Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 4602052, 8 pages
https://doi.org/10.1155/2019/4602052

Research Article

Interactive Q-Learning Approach for Pick-and-Place Optimization of the Die Attach Process in the Semiconductor Industry

Gilseung Ahn,[1] Myunghwan Park,[1] You-Jin Park,[2] and Sun Hur [1]

[1] *Department of Industrial and Management Engineering, Hanyang University, Ansan 15588, Republic of Korea*
[2] *Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei, Taiwan*

Correspondence should be addressed to Sun Hur; hursun@hanyang.ac.kr

Received 13 November 2018; Revised 11 January 2019; Accepted 6 February 2019; Published 18 February 2019

Guest Editor: Anders E. W. Jarfors

In semiconductor back-end production, the die attach process is one of the most critical steps affecting overall productivity. Optimization of this process can be modeled as a pick-and-place problem known to be NP-hard. Typical approaches are rule-based and metaheuristic methods. The two have high or low generalization ability, low or high performance, and short or long search time, respectively. The motivation of this paper is to develop a novel method involving only the strengths of these methods, i.e., high generalization ability and performance and short search time. We develop an interactive Q-learning in which two agents, a pick agent and a place agent, are trained and find a pick-and-place (PAP) path interactively. From experiments, we verified that the proposed approach finds a shorter path than the genetic algorithm given in previous research.

1. Introduction

The entire semiconductor manufacturing process can largely be divided into two sequential subprocesses of front-end production and back-end production. Both production processes involve many complicated steps for wafer fabrication, probe testing and sorting, assembly, final testing, etc. [1]. The front-end process refers to wafer fabrication (fab) and a wafer probe test called electrical die sorting (EDS), whereas the back-end process refers to preassembly, packaging (assembly), and burn-in and functional tests of individual semiconductor chips. Once the front-end processes are completed, the wafers are transferred to back-end production to facilitate their integration into electronic devices and to undergo a final performance test. Particularly, in back-end production, based on quality and location information of individual chips (die) derived from the EDS test, only good semiconductor chips are individually picked and attached to the support structure (e.g., the lead frame) on a strip by an automatic robot arm. This process is called the die attach process [2]. In semiconductor back-end production, this process is regarded as one of the most critical steps since it is the first packaging layer in contact with the die, which demands a high level of operational precision. Thus, optimization of the die attach process is important to maximize the overall productivity in semiconductor back-end production.

Optimization of the die attach process can be formulated as a typical pick-and-place (PAP) problem. The PAP problem is to find the shortest path to pick every component (good dies) and place it on the plate (strip). Because it is a well-known NP-hard problem [3], heuristic approaches such as the rule-based and metaheuristic methods are usually adopted. The rule-based approach is to define a proper dispatching rule and find the shortest path using the predefined rule. Park et al. [1] introduced 11 rules for PAP of the die attach process, which are categorized according to direction and starting point (e.g., the counterclockwise path starting from the center). Huang et al. [4] applied a greedy rule, which is to pick the nearest component and place it on the nearest plate, to solve a PAP problem of multirobot coordination. The metaheuristic approach is to design a metaheuristic algorithm, such as genetic algorithm (GA), particle swarm optimization (PSO), and Tabu search, and apply it to solve PAP problems. Park et al. [1] developed GA, which uses

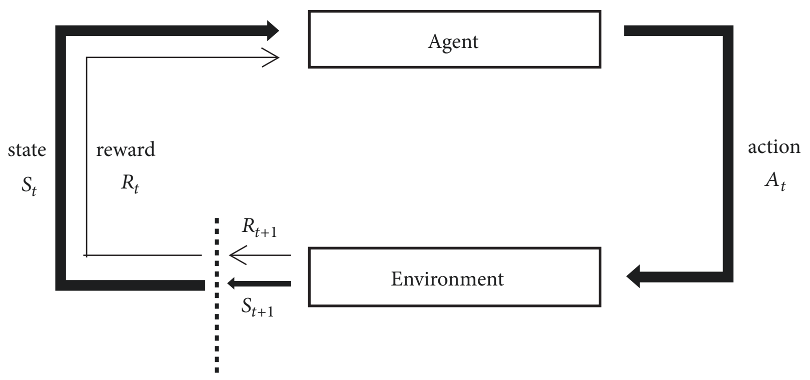


FIGURE 1: Process of reinforcement learning.

a binary matrix as an encoding scheme and batch and row-string crossover operations. Torabi et al. [5] solved a nozzle selection (pick) and component allocation (place) problem to minimize the workload of the bottleneck while maximizing all corresponding appropriateness factors by a multiobjective particle swarm optimization. Liu and Kozan [6] proposed a blocking job shop scheduling problem with robotic transportation and developed a hybrid algorithm based on a Tabu search.

Each approach can be evaluated by generalization ability, performance, and search time. The generalization ability reveals how easily the approach can be applied to solve various PAP problems once the model including rules and algorithms is defined or developed. The distance of a path determines the performance of the approach. Finally, the search time is the time to find a path. Rule-based approach is better than metaheuristic approach in terms of generalization ability and search time. That is, predefined rules can be applied to solve any PAP problems in a short time, but the metaheuristic approach designed for a problem cannot be applied to different ones and requires relatively long time to solve a problem. However, the performance of metaheuristic approach is better than that of rule-based approach because the path yielded from metaheuristic is usually shorter than that from rule-based approach.

Reinforcement learning explores the space of feasible solutions efficiently and effectively [7] and therefore has been actively applied to solve various optimization problems as alternatives of metaheuristic and rule-based approaches [8–14]. For example, Dou et al. [8] proposed a path planning method for mobile robots in intelligent warehouses based on Q-learning. In this method, the reward is designed to encourage fewer steps of the robots. As another example, Shiue et al. [9] applied a reinforcement learning approach to solve real-time scheduling problems in a smart factory, where Q-learning is used to determine the dispatching rule given remaining jobs.

The objective of this paper is to develop an approach that overcomes the weak points of the typical approach but incorporates the strengths. That is, this approach can show high performance and generalization capability and a shorter search time. With this motivation, we develop an interactive Q-learning method including two interactive agents. That is, a pick agent and a place agent transfer information, restrict or encourage certain actions, and have an impact on the training process of the other. This approach shows higher performance than the metaheuristic approach, relatively shorter training time, and a similar search time to the rule-based approach.

The rest of this paper is organized as follows. Section 2 briefly explains reinforcement learning and Q-learning, which is the basis of our approach. Section 3 states the PAP problem in a die attach process and develops a mathematical model. Section 4 develops the interactive Q model, and Section 5 compares the developed model and metaheuristic approach. Finally, Section 6 concludes the paper and suggests a future research direction.

2. Reinforcement Learning

Reinforcement learning is to train an agent to discover a sequence of actions that yields the highest reward by searching for many pairs of states and actions [15], as presented in Figure 1.

To be more specific, when the system stays at state S_t and an agent performs action A_t, the environment returns reward R_{t+1}, and the system transfers to the new state S_{t+1}. An episode is the sequence of an agent's actions starting from initial state S_0 to terminal state S_T. The purpose of the reinforcement learning is to find the action sequence $A_1^* - A_2^* - \cdots - A_T^*$ (or state sequence $S_1^* - S_2^* - \cdots - S_T^*$) with the trained agent experiencing many episodes.

Among the reinforcement learning methods, Q-learning selects an action on a state in a greedy manner with regard to Q-function. That is, the action achieving the maximum Q-function on a state is selected. Q-function for a pair of state and action (S, A) is defined as the expected sum of discounted rewards when performing A on state S [16]. Q-table is a matrix whose (i, j)th component is the Q-function value of the i^{th} state and the j^{th} action, (S_i, A_j).

The Q-table is updated through episodes from the (old) Q-table by the Bellman equation [17]:

$$\widetilde{Q}(S_t, A_t) = (1-\alpha)\cdot Q(S_t, A_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_A \widetilde{Q}(S_{t+1}, A)\right), \quad (1)$$

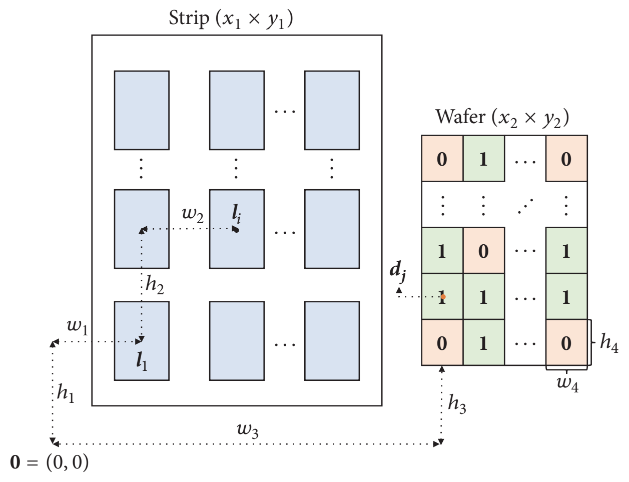


FIGURE 2: Coordinate information of the strip and wafer.

where t is time period of an episode; $\widetilde{Q}(S_t, A_t)$ and $Q(S_t, A_t)$ are the updated Q-function and old Q-function for S_t and A_t, respectively; $0 < \alpha \leq 1$ is the learning rate; $0 \leq \gamma \leq 1$ is the discount rate; and r_t is the reward when performing A_t on S_t. In this equation, $\widetilde{Q}(S_t, A_t)$ is a weighted sum of the old value and the learned value, where the weights are obtained from the learning rate. The discount rate γ reflects the importance of the expected value of the next action $\max_A \widetilde{Q}(S_{t+1}, A)$.

3. Problem Statement and Mathematical Model

3.1. Problem Statement. In postfabrication process of the semiconductor manufacturing, the probe test is performed on individual die (chip) of wafers to identify defect of dies and classify dies by the level of its quality. Then, the only “good” dies (i.e., conforming chips) are assembled and packaged in the assembly step in back-end production [18, 19]. After the test, a robot arm repeatedly picks up a good die in a wafer and places it on a lead frame in a strip until every good die is transferred in the die attach process. The robot arm starts and ends the process at the origin. That is, the start point and end point are the same and fixed. The total moving distance highly depends on the order of pick and place (i.e., path). Therefore, the considered problem is to find the shortest path to transfer every good die to a lead frame with the robot arm. Note that the Manhattan distance metric is employed to calculate distance, because the robot arm moves only vertically or horizontally due to several technical issues.

We assume that the strip and wafer are located at the left and right sides on the coordinate, respectively, and are rectangular, as presented in Figure 2. Even though these assumptions may be unrealistic (e.g., the wafer is round), they do not affect the modeling and development of the interactive Q-learning approach.

3.2. Notations. Notations used in this paper are as follows.

Indices

i: Lead frame index, $(i = 1, 2, \ldots, n)$

j: Good die index, $(j = 1, 2, \ldots, m)$

z: Agent index, ($z = 1$: pick agent, $z = 2$: place agent)

Problem Parameters

(x_1, y_1): Strip shape

(x_2, y_2): Wafer shape

$\boldsymbol{l}_i$: Coordinate of lead frame i

$\boldsymbol{d}_j$: Coordinate of good die j

w_1: Horizontal distance between the origin and $\boldsymbol{l}_1$

h_1: Vertical distance between the origin and $\boldsymbol{l}_1$

w_2: Horizontal distance between the two consecutive lead frames

h_2: Vertical distance between the two consecutive lead frames

w_3: Horizontal distance between the origin and the first die

h_3: Vertical distance between the origin and the first die

w_4: Horizontal distance between two consecutive dies

h_4: Vertical distance between two consecutive dies

Model Parameters

T: Maximum number of iterations

Ω^z: State space of agent z

S^z: Current state of agent z

A^z: Action space of agent z

$\mathcal{F}^z$: Feasible action space of agent z

Q^z: Q-table of agent z

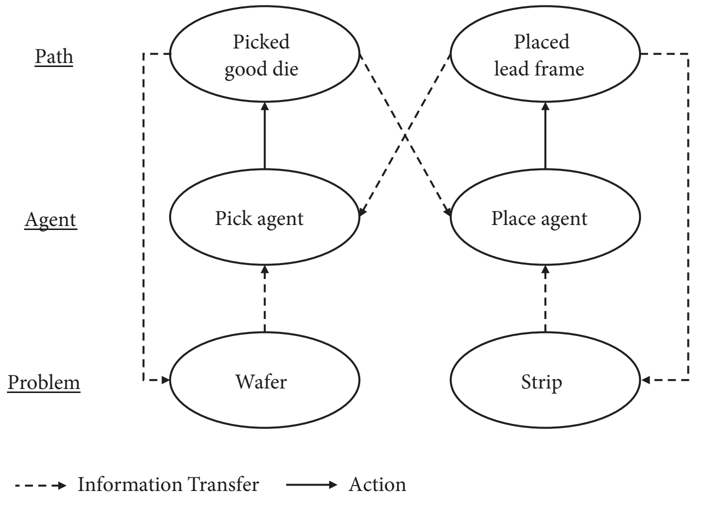


FIGURE 3: Interactive Q Model configuration.

P: Path, $P = P^0 - P^1 - P^2 - \cdots - P^{|P|} - P^{-1}$, where P^0 and P^{-1} are the starting and ending points of the path, respectively. P^n is the n^{th} point (good die if n is odd and lead frame otherwise) and $|P|$ is the length of the path

$dist(P)$: Total distance of path P

3.3. Mathematical Model. The mathematical model is presented as follows:

$$\text{Minimize} \quad dist(P), \tag{2}$$

$$\text{Subject to} \quad \bigcup_{k=1}^{|P|/2} P^{2k-1} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_m\}, \tag{3}$$

$$P^{2k} \in \{\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n\}, \quad \text{for } k = 1, 2, \ldots, \frac{|P|}{2}, \tag{4}$$

$$\bigcup_{k=n\times t+1}^{n\times(t+1)} P^{2k} = \{\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n\}, \quad \text{for } t = 0, 1, \ldots, \left[\frac{m}{n}\right] \tag{5}$$

$$\bigcup_{k=n\times([m/n]+1)}^{|P|/2} P^{2k} \in \{\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n\}. \tag{6}$$

The objective function in (2) is to minimize the total distance, where

$$dist(P) = \left(P_x^1 + P_y^1\right) + \sum_{k=1}^{|P|/2} \left(\left|P_x^{2k-1} - P_x^{2k}\right| + \left|P_y^{2k-1} - P_y^{2k}\right|\right) + \left(P_x^{|P|} + P_y^{|P|}\right), \tag{7}$$

where (P_x^{2k-1}, P_y^{2k-1}) and (P_x^{2k}, P_y^{2k}) are the coordinates of the k^{th} selected good die and lead frame, respectively. $(P_x^1 + P_y^1) = |P_x^1 - 0| + |P_y^1 - 0|$ denotes the distance from the origin to the first selected good die, and $(P_x^{|P|} + P_y^{|P|}) = |0 - P_x^{|P|}| + |0 - P_y^{|P|}|$ denotes the distance from the last selected lead frame to the origin.

Constraints (3) and (4) indicate that the robot arm picks a good die and places it on the empty lead frame. Constraints (5) and (6) show that the good dies are transferred until an empty strip is full; once the strip is full, it is replaced with a new one until every good die is transferred. Two typical example paths satisfying these constraints when $m = 5$ and $n = 2$ are $\mathbf{0} - \boldsymbol{d}_1 - \boldsymbol{l}_1 - \boldsymbol{d}_2 - \boldsymbol{l}_2 - \boldsymbol{d}_3 - \boldsymbol{l}_1 - \boldsymbol{d}_4 - \boldsymbol{l}_2 - \boldsymbol{d}_5 - \boldsymbol{l}_2 - \mathbf{0}$ and $\mathbf{0} - \boldsymbol{d}_3 - \boldsymbol{l}_2 - \boldsymbol{d}_2 - \boldsymbol{l}_1 - \boldsymbol{d}_3 - \boldsymbol{l}_1 - \boldsymbol{d}_1 - \boldsymbol{l}_2 - \boldsymbol{d}_5 - \boldsymbol{l}_1 - \mathbf{0}$. As one can see, the path starts and ends at the origin, there is no duplicate of good dies in a path, and a new strip is installed only when the previous strip is full.

4. Interactive Q Model

4.1. Configuration and Deployment. The proposed interactive Q model is configured as shown in Figure 3.

As seen in Figure 3, the model consists of three layers: problem, agent, and path. The problem layer transfers the information of the problem (i.e., number and location of the remaining good dies in the wafer and empty lead frames in the strip) to the agent layers. The pick and place agents in the agent layer select a good die to be picked and a lead frame to which the good die is placed considering the information, respectively. The selected good die and lead frame are appended to the path. Finally, the information that the selected good die and lead frame are no longer feasible is transferred to the problem layer.

Agent $z \in \{1, 2\}$ consists of state space Ω^z, action space A^z, feasible action space $\mathcal{F}^z$, and Q-table Q^z. $\Omega^1 = \{\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n\}$ and $\Omega^2 = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_m\}$ are sets of locations of the robot arm before picking and placing, respectively. $A^1 = \{a_1^1, a_2^1, \ldots, a_m^1\}$ and $A^2 = \{a_0^2, a_1^2, a_2^2, \ldots, a_n^2\}$ are action

```
Input: $\Omega^1, \Omega^2, A^1, A^2, Q^1, Q^2$
Procedure: Initialize $\mathscr{F}^1 = A^1$ and $\mathscr{F}^2 = A^2$
           Initialize $P^0 = \mathbf{0}$ and $P^{2m+2} = \mathbf{0}$
           Initialize $S^1 = \mathbf{0}$ # $S^1$: current state of the pick agent
           Initialize $p = 0$
           Until $\mathscr{F}^1 = \emptyset$ do {
               Increase $p$ by 1
               If $\mathscr{F}^2 = \emptyset$ do {# If a strip is full, the strip is replaced with a new one
                   $\mathscr{F}^2 = A^2$}
               # Pick agent procedure
               $a^1_{j^*} = \mathrm{argmax}_{a^1_j \in \mathscr{F}^1}(Q^1_{S^1,j})$ # Select an action of the pick agent
               $P^p = \boldsymbol{d}_{j^*}$
               $S^2 = \boldsymbol{d}_{j^*}$ # Update the current state of the place agent
               $\mathscr{F}^1 = \mathscr{F}^1 - \boldsymbol{d}_{j^*}$ # Update the feasible action space of the pick agent
               # Place agent procedure
               Increase $p$ by 1
               $a^2_{i^*} = \mathrm{argmax}_{a^2_i \in \mathscr{F}^2}(Q^2_{S^2,i})$ # Select an action of the place agent
               $P^p = \boldsymbol{l}_{i^*}$
               $S^1 = \boldsymbol{l}_{i^*}$
               $\mathscr{F}^2 = \mathscr{F}^2 - \boldsymbol{l}_{i^*}$ # Update the feasible action space of the place agent
               }
Output: $P = P^0 - P^1 - P^2 - \cdots - P^{|P|} - P^0$
```

Algorithm 1: Routing algorithm of the interactive Q model.

spaces of the pick agent and place agent, respectively, where a^1_j is to pick a good die j, and a^2_i is to place the selected good die on lead frame i. Here, a^2_0 indicates that the robot arm returns to the origin. Feasible actions of the pick agent $\mathscr{F}^1 \subset A^1$ and place agent $\mathscr{F}^2 \subset A^2$ include good dies in a wafer and empty lead frames, respectively. That is, good dies in a wafer and empty lead frames are feasible. $Q^1_{i,j}$ indicates a Q-function of a^1_j when the current state is $\boldsymbol{l}_i$ for $i = 0, 1, \ldots, n$ and $j = 1, \ldots, m$, where $\boldsymbol{l}_0$ is the origin. Likewise, $Q^2_{j,i}$ indicates a Q-function of a^2_i when the current state is $\boldsymbol{d}_j$ for $i = 1, \ldots, n$ and $j = 0, 1, \ldots, m$, where $\boldsymbol{d}_0$ is the origin.

Pick and place agents act in a greedy manner with respect to Q-table, whose training method is explained in Section 4.2. The pick agent picks a good die with the maximum Q-value in a wafer (i.e., it selects the action with the maximum Q-value among feasible actions) when the current state is $\boldsymbol{l}_i$, as follows:

$$a^1_{j^*} = \underset{a^1_j \in \mathscr{F}^1}{\mathrm{argmax}} \left(Q^1_{i,j}\right), \tag{8}$$

where $a^1_{j^*}$ is the selected action of the pick agent. The place agent places the selected good die j on an empty lead frame with the maximum Q-value, as follows:

$$a^2_{i^*} = \underset{a^2_i \in \mathscr{F}^2}{\arg\max} \left(Q^2_{j,i}\right), \tag{9}$$

where $a^2_{i^*}$ is the selected action of the place agent. After the actions are determined, the feasible action space is updated.

Algorithm 1 shows the routing process of the interactive Q model.

4.2. Training Algorithm. Q^1 and Q^2 are updated using the Bellman equation presented in (1). When updating the Q-tables, they impact one other because the estimated optimal future values (i.e., $\max_{a^2_k \in \mathscr{F}^2}(\widetilde{Q}^2_{j,k})$ and $\max_{a^1_k \in \mathscr{F}^1}(\widetilde{Q}^1_{i,k})$) of the pick and place agents are the Q-function values of the place and pick agents, as given in (10) – (11).

$$\widetilde{Q}^1_{i,j} = (1 - \alpha) \times Q^1_{i,j} + \alpha \times \left(r^1_{i,j} + \gamma \times \max_{a^2_k \in \mathscr{F}^2} \left(\widetilde{Q}^2_{j,k}\right) \right), \tag{10}$$

$$\widetilde{Q}^2_{j,i} = (1 - \alpha) \times Q^2_{j,i} + \alpha \times \left(r^2_{j,i} + \gamma \times \max_{a^1_k \in \mathscr{F}^1} \left(\widetilde{Q}^1_{i,k}\right) \right), \tag{11}$$

where $\widetilde{Q}^1_{i,j}$ and $\widetilde{Q}^2_{j,i}$ are updated (new) Q-functions, while $Q^1_{i,j}$ and $Q^2_{j,i}$ are old Q-functions. $r^1_{i,j}$ is the reward of the pick agent for selecting a^1_j from $\boldsymbol{l}_i$, which is computed as the reciprocals of distances $\boldsymbol{l}_i$ and $\boldsymbol{d}_j$. $r^2_{j,i}$ is also similarly defined and computed. Note that $Q^1_{i,j}$ and $Q^2_{j,i}$ are updated only when a^1_j and a^2_i are selected when the current states are $\boldsymbol{l}_i$ and $\boldsymbol{d}_j$, respectively.

Algorithm 2 shows the training algorithm for Q^1 and Q^2 based on general Q-learning's exploration strategy, considering both current and future reward. To be more concrete, Q-value of agent z ($z = 1, 2$) is updated by taking the weighted average of the current reward $Q^z_{S^z,j^*}$ and future reward $r^z_{S^1,j^*} + \gamma \times \max_{a^{3-z}_k \in \mathscr{F}^{3-z}}(\widetilde{Q}^2_{j^*,k})$, which means that each agent whose current state is S_t at time t considers distance between S_t and S_{t+1} as well as between S_{t+1} and S_{t+2}. It trains Q-tables using the wafer including no bad dies, but the trained Q-tables can be used to solve every problem if the wafer and strip

```
**Input:** $\Omega^1, \Omega^2, A^1, A^2, \mathrm{T}$
**Procedure:** Initialize every element in $\widetilde{Q}^1, Q^1, \widetilde{Q}^2, Q^2$ with arbitrary numbers
    Initialize $t = 1$
    **Until** $(\widetilde{Q}^1 = Q^1$ and $\widetilde{Q}^2 = Q^2)$ **or** $t = \mathrm{T}$ **do** {
        $\widetilde{Q}^1 = Q^1$
        $\widetilde{Q}^2 = Q^2$
        Initialize $\mathcal{F}^1 = A^1$ and $\mathcal{F}^2 = A^2$
        Initialize $S^1 = \mathbf{0}$ # $S^1$: current state of the pick agent
        **Until** $\mathcal{F}^1 = \emptyset$ **do** {
            **If** $\mathcal{F}^2 = \emptyset$ **do** {# If a strip is full, it is replaced with a new one
                $\mathcal{F}^2 = A^2$}
            # Pick agent update
            $a^1_{j*} = \mathrm{argmax}_{a^1_j \in \mathcal{F}^1}(Q^1_{S^1,j})$
            $\widetilde{Q}^1_{S^1,j*} = (1-\alpha) \times Q^1_{S^1,j*} + \alpha \times (r^1_{S^1,j*} + \gamma \times \max_{a^2_k \in \mathcal{F}^2}(\widetilde{Q}^2_{j*,k}))$
            $S^2 = \boldsymbol{d}_{j*}$ # Update the current state of the place agent
            $\mathcal{F}^1 = \mathcal{F}^1 - \boldsymbol{d}_{j*}$ # Update the feasible action space of the pick agent
            # Place agent update
            $a^2_{i*} = \mathrm{argmax}_{a^2_i \in \mathcal{F}^2}(Q^2_{S^2,i})$ # Select the action of the place agent
            $\widetilde{Q}^2_{S^2,i*} = (1-\alpha) \times Q^2_{S^2,i*} + \alpha \times (r^2_{S^2,i*} + \gamma \times \max_{a^1_k \in \mathcal{F}^1}(\widetilde{Q}^1_{i*,k}))$
            $S^1 = \boldsymbol{l}_{i*}$
            $\mathcal{F}^2 = \mathcal{F}^2 - \boldsymbol{l}_{i*}$ # Update the feasible action space of the place agent
        Increase $t$ by 1
        }
**Output:** $\widetilde{Q}^1, \widetilde{Q}^2$
```

Algorithm 2: Q-tables update algorithm.

shapes are the same. Thus, the interactive Q-model has higher generalization capability than metaheuristic algorithms.

5. Experiment

In this section, we compare the performance of our interactive Q-learning model with those of the GA model and rule-based models presented in the literature [1].

5.1. Parameters. The problem parameters are obtained from Park et al. [1] as follows: $(x_1, y_1) = (10, 4)$, $(x_2, y_2) = (9, 9)$, $w_1 = 14$, $w_2 = 12$, $w_3 = 66$, $w_4 = 8$, $h_1 = 22$, $h_2 = 20$, $h_3 = 18$, $h_4 = 4$.

Four wafers are considered according to the yield rates of dies (80% and 90%) and the bad die distribution (bivariate normal and uniform distribution). The wafers are depicted in Figure 4, where colored cells denote bad dies.

Four rules are adopted from Park et al. [1], where they are defined as follows.

Rule 1. The robot arm moves good dies from the upper-left side of a wafer to empty lead frames in the upper-left side of a strip.

Rule 2. The robot arm starts moving the good dies from the upper-left side of a wafer to empty lead frames in the upper-right side of a strip.

Rule 3. The robot arm moves good dies from the upper-right side of a wafer to empty lead frames in the upper-right side of a strip.

Rule 4. The robot arm starts moving the good dies from the upper-right side of a wafer to empty lead frames in the upper-left side of a strip.

The parameters of GA were set as follows: (1) the number of initial solutions: 200, (2) the number of iterations: 100, and (3) crossover operation: batch and row-string crossover operation. The interactive Q-learning model's parameters are set as $\alpha = 0.01$, $\gamma = 0.99$, and $\mathrm{T} = 1000$, and the model is trained using the same sized wafer and strip. Note that the wafer for training is assumed to have no bad die, and the trained model can be applied to every problem if the shapes of the wafer and strip remain fixed.

5.2. Results. Figure 5 shows the performance comparison results.

Figure 5 shows that the proposed model outperforms the GA and rule-based models in every problem, implying that the model has sufficient generalization ability, as well as excellent performance. Specifically, 13.14%, 13.23%, 7.05%, and 8.09% of the total distance are decreased from GA for wafers A, B, C, and D, respectively. All rule-based models have produced longer total distances than GA for all wafers. From the experiment result, we can conclude that (1) the interactive Q model outperforms GA, (2) it is more effective when the yield rate is low or when the defective rate is high, and (3) it is robust to the bad die distribution.

As for the running time, the proposed model and rule-based models take less than a second, but GA takes more than an hour to yield a path and is highly dependent on the number

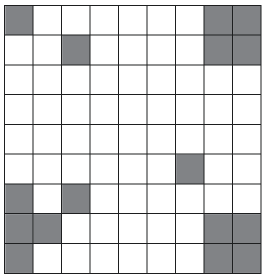
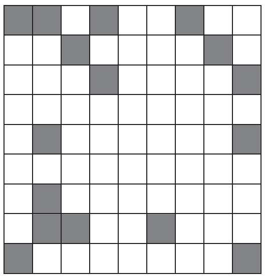
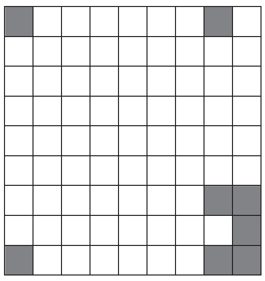
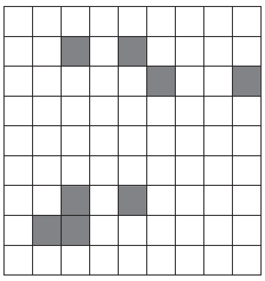
Yield rate	Bad die distribution	
	Bivariate normal distribution	Uniform distribution
80%	 Wafer A	 Wafer B
90%	 Wafer C	 Wafer D

Figure 4: Wafers for the experiments.

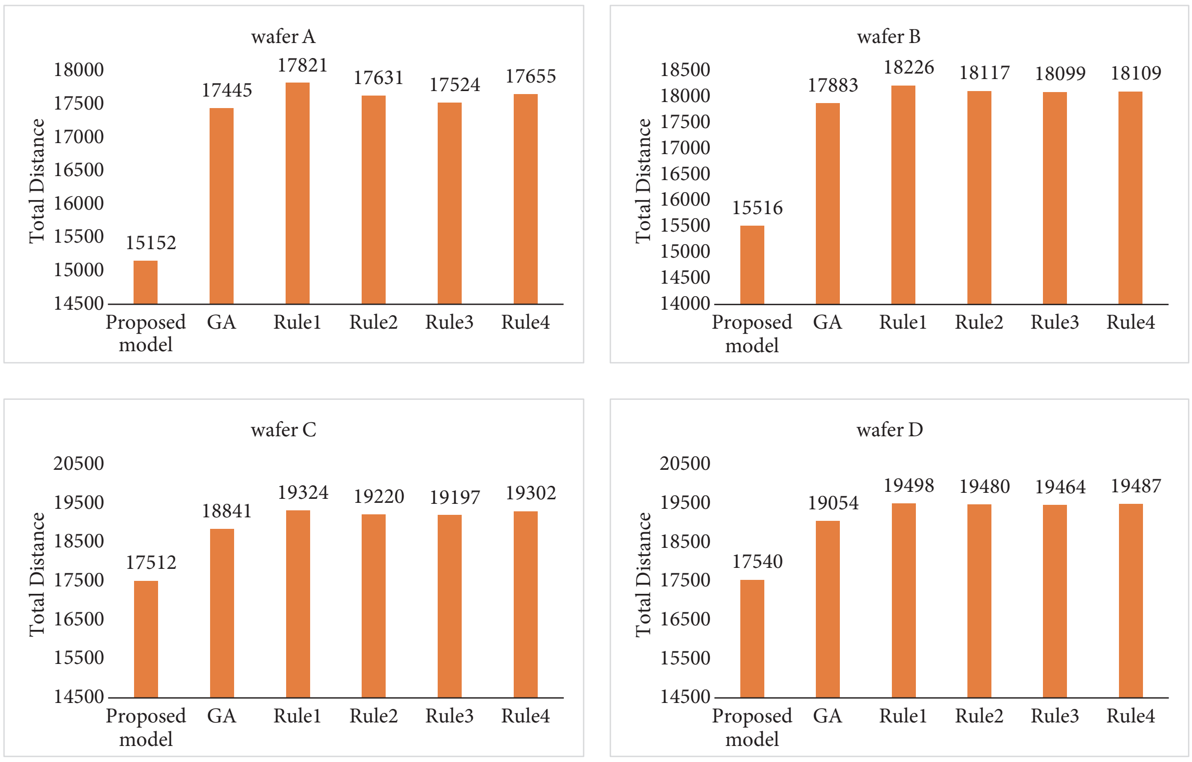


Figure 5: Comparison results of the interactive Q model, GA, and rules.

of iterations and initial solutions. In addition, the interactive Q model requires short training time. In our experiment, the training time when $(x_1, y_1) = (10, 4)$ and $(x_2, y_2) = (9, 9)$ is 254 seconds. That is, one episode requires only 0.254 seconds. In addition, yielding a path by means of the model takes less than 1 second. This may imply that the model can be trained and applied to solve the problem in real time.

6. Conclusion

In this paper, we addressed the PAP problem of the die attach process to maximize the overall productivity in semiconductor back-end production. Due to the NP-hardness of this problem, rule-based and metaheuristic approaches have been applied. These approaches, however, should be improved because the metaheuristic approach has low generalization ability. With this motivation, we developed an interactive Q-learning approach equipped with two interactive agents to find a path. The experiment revealed that the proposed model shows higher performance than GA and has almost the same search time as the rule-based approach.

In future work, one can modify our interactive Q-learning to solve various optimization problems in the semiconductor industry and PAP problems in other industries. In addition, a hybrid approach of Q-learning and other approaches such as rule-based and metaheuristic approaches can be developed to solve more complicated problems efficiently. That is, incorporating the proposed interactive Q-learning with rule-based approach can increase the generalization ability, which will solve PAP problems with different wafers.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2017R1A2B4006643).

References

[1] Y.-J. Park, G. Ahn, and S. Hur, “Optimization of pick-and-place in die attach process using a genetic algorithm,” *Applied Soft Computing*, vol. 68, pp. 856–865, 2018.

[2] G. Weigert, A. Klemmt, and S. Horn, “Design and validation of heuristic algorithms for simulation-based scheduling of a semiconductor backend facility,” *International Journal of Production Research*, vol. 47, no. 8, pp. 2165–2184, 2009.

[3] A. Srivastav, H. Schroeter, and C. Michel, “Approximation algorithms for pick-and-place robots,” *Annals of Operations Research*, vol. 107, no. 1-4, pp. 321–338, 2001.

[4] Y. Huang, R. Chiba, T. Arai, T. Ueyama, and J. Ota, “Robust multi-robot coordination in pick-and-place tasks based on part-dispatching rules,” *Robotics and Autonomous Systems*, vol. 64, pp. 70–83, 2015.

[5] S. A. Torabi, M. Hamedi, and J. Ashayeri, “A new optimization approach for nozzle selection and component allocation in multi-head beam-type SMD placement machines,” *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 700–714, 2013.

[6] S. Q. Liu and E. Kozan, “A hybrid metaheuristic algorithm to optimise a real-world robotic cell,” *Computers & Operations Research*, vol. 84, pp. 188–194, 2017.

[7] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[8] J. Dou, C. Chen, and P. Yang, “Genetic scheduling and reinforcement learning in multirobot systems for intelligent warehouses,” *Mathematical Problems in Engineering*, vol. 2015, Article ID 597956, 10 pages, 2015.

[9] S. Yeou-Ren, K. C. Lee, and C. T. Su, “Real-time scheduling for a smart factory using a reinforcement learning,” *Computers & Industrial Engineering*, vol. 125, pp. 604–614, 2018.

[10] S. Li, L. Ding, H. Gao, C. Chen, Z. Liu, and Z. Deng, “Adaptive neural network tracking control-based reinforcement learning for wheeled mobile robots with skidding and slipping,” *Neurocomputing*, vol. 283, pp. 20–30, 2018.

[11] A. S. Xanthopoulos, A. Kiatipis, D. E. Koulouriotis, and S. Stieger, “Reinforcement learning-based and parametric production-maintenance control policies for a deteriorating manufacturing system,” *IEEE Access*, vol. 6, pp. 576–588, 2017.

[12] J. Shahrabi, M. A. Adibi, and M. Mahootchi, “A reinforcement learning approach to parameter estimation in dynamic job shop scheduling,” *Computers & Industrial Engineering*, vol. 110, pp. 75–82, 2017.

[13] A. Kara and I. Dogan, “Reinforcement learning approaches for specifying ordering policies of perishable inventory systems,” *Expert Systems with Applications*, vol. 91, pp. 150–158, 2018.

[14] R. F. Atallah, C. M. Assi, and J. Y. Yu, “A reinforcement learning technique for optimizing downlink scheduling in an energy-limited vehicular network,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 4592–4601, 2017.

[15] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, MIT Press, 2018.

[16] M. K. Mainali, K. Shimada, S. Mabu, and K. Hirasawa, “Optimal route based on dynamic programming for road networks,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 12, no. 6, pp. 546–553, 2008.

[17] L. Buşoniu, R. Babuška, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[18] K. Nakamae, A. Itoh, and H. Fujioka, “Fail pattern classification and analysis system of memory fail bit maps,” in *Proceedings of the 2001 International Conference on Modeling and Simulation of Microsystems - MSM 2001*, pp. 598–601, March 2001.

[19] C.-H. Wang, “Recognition of semiconductor defect patterns using spatial filtering and spectral clustering,” *Expert Systems with Applications*, vol. 34, no. 3, pp. 1914–1923, 2008.