An MDP-based lifter assignment algorithm for inter-floor transportation in semiconductor fabrication

Kyohong Shin¹ · Hoon Jang² · Haejoong Kim³

Received: 25 May 2021 / Accepted: 13 November 2021 / Published online: 21 January 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2021, corrected publication 2022

Abstract
As semiconductor device geometries continue to shrink, the semiconductor manufacturing process becomes increasingly complex. This usually results in unbalanced utilization of machines and decreases overall productivity. One way to resolve such a problem is to share the manufacturing resources between different lines divided by floors. To this end, designing an efficient lifter assignment method to more efficiently manage transfer requests (TRs) of wafer lots to different floors is required. Motivated by this, our study addresses the assignment of lifters for delivering wafer lots to different floors. In contrast to previous studies that consider only the current state of the system, our approach considers both the current and possible future states of the system in a probabilistic manner in the Markov decision process. To overcome the curse of dimensionality of the original problem, we design an efficient algorithm using clustering, partitioning, and tournament methods. Experiments based on historical data confirm the effectiveness of the proposed algorithm in reducing transportation times and delivery delays compared to the benchmark rules in practice and the method in the state-of-the-art. Sensitivity analysis demonstrates the robustness of the proposed model as the number of TRs increased. The proposed approach is expected to yield significant economic savings in both operating costs and labour.

Keywords Manufacturing · Semiconductor · Automated material handling system · Lifter · Markov decision process

1 Introduction
Wafer fabrication in semiconductor manufacturing is an extremely complex process in which wafers are transported through hundreds of steps. To manage these complex material flows, automated material handling systems (AMHSs), sophisticated material control systems that move materials from one machine to another have been successfully applied in a semiconductor fabrication facility (FAB). In particular, an AMHS employing an overhead hoist transfer (OHT) reduces wafer transport time dramatically and contributes to reduced cycle times and increased machine utilization by improving on time delivery.

As semiconductor manufacturing processes are becoming increasingly complex, and the number of production steps is increasing, a single floor line often has an inadequate capacity to cover such production needs. Therefore, FABs with multiple floors have been introduced recently [1]. At the initial FAB operation phase, the inter-floor transportation from one floor to another is constrained at a minimum level because the capacity of the process machines of each floor is kept balanced. However, as the product mix changes, the equipment utilization on each floor becomes unbalanced and more inter-floor transportation to share equipment resources on other floors is required. In general, the number of lifters for inter-floor transportation is determined at the minimum level because the lifter itself is expensive to set up and consumes cleanroom space. Thus, the lifters become a major bottleneck of the AMHS when the inter-floor transportation increases [2].

The typical process of inter-floor transportation is shown in Fig. 1a. Upon the arrival of a transfer request (TR) to
transport a wafer lot from a machine on floor A to another on floor B, the AMHS allocates an OHT and a lifter connected to floor B. Then, the assigned OHT unloads the wafer lot from the source machine and moves it to the dedicated lifter. After the OHT arrives at the lifter, it loads the wafer lot into the in-buffer port of the lifter for transportation to floor B. The buffer port is where a lot waits for transport to another floor by the lifter or to the destination by an OHT afterward. The wafer lot in the in-buffer port is transported to the out-buffer port on another floor by the rack-master (Fig. 1b). Finally, an OHT on floor B transports the wafer lot to the destination machine.

Figure 2 shows states and times of a wafer lot, two OHTs (both in floor A and B), and a rack-master according to the sequence of events, from the perspective of the departure floor (Floor A). In this situation, relevant times are defined as follows:

- Assign time: the time from the moment of the TR's arrival until the OHT arrives at the source machine
- Travel time: the time difference between unloading the lot from the source and loading it into the destination
- Waiting time: the time required for the wafer lot to be picked up by the rack-master after being loaded into the lifter's buffer port
• Cycle time: the time difference between the moment the rack-master moves to pick up the lot and place it down on the other floor port

Transportation times between floors are significantly longer than on a single floor because inter-floor transportation consists of two transfers by OHTs and one transfer using a lifter. Thus, when the lifter is not selected properly, the travel time to the destination machine could be excessively long, and the shared machines would not be fully utilized on account of the transportation delays [2]. In practice, when the AMHS selects the lifter for each inter-floor transportation, dispatching rules, such as the round-robin rule or the shortest-transfer-distance rule, are widely used instead of analytical models to identify the acceptable solutions in the dynamic environment of the actual scenario. The round-robin rule chooses lifters in a pre-set circular order; TRs are evenly distributed to all lifters. This approach has the advantage of keeping lifter workloads even and minimizing the variance of the waiting time in the buffer port of the lifter. It also decreases the vehicle congestion around lifters by preventing excessive traffic on specific lifters. However, because the travel time of an OHT is not considered, a lifter with a long distance to travel is frequently assigned, which causes a long travel time and the vehicle congestion. The shortest-transfer-distance rule assigns wafer lots to the nearest lifters to reduce the travel time. However, TRs could then be concentrated on specific lifters, which increase waiting time in the lifters. Therefore, undesirable situations, such as delays in transporting a wafer lot to another line, cannot be prevented. The trade-off exists between the travel time on the source floor and waiting time at the lifter. The current practice could not consider this trade-off because the future states of lifters operating in a FAB are not considered whatsoever.

We propose a method based on the Markov decision process (MDP) to efficiently assign lifters to minimize the inter-floor transportation time under dynamic manufacturing conditions. We consider future states of the system in a probabilistic manner and intend to minimize the long-run expected average transportation time. We also evaluate the performance of the proposed approach using an emulator with data gathered from an actual FAB.

The remainder of this paper is structured as follows. Section 2 reviews the existing literature related to this study and discusses its implications and significance. Our contributions to the existing literature are also clarified. Section 3 details the mathematical model. Section 4 describes the proposed solution algorithm based on MDP. Section 5 describes the experimental design, the results of the experiments, and their implications. Finally, Section 6 concludes our study.

2 Literature review

Due to the large volume of related literature, we first briefly review the research stream of AMHS, and subsequently focus on studies concentrating on a lifter-assignment problem. For a general review on AMHS, refer to [3–6].

Many studies on vehicle management, including vehicle allocation, dispatching, and routing, which are viable options to improve productivity without the power shut down of FABs, have been conducted. The vehicle allocation problem in the semiconductor AMHS generally involves determining the optimal fleet size to minimize transportation time. Many researchers have proposed determination of the optimal fleet size, which would fulfill a specific transfer requirement [7–10]. To respond to highly dynamic manufacturing environments, some studies focused on dynamically repositioning idle vehicles to appropriate locations to minimize transport time [11–14]. Another important issue in managing AMHS is to identify the appropriate dispatching rules of vehicles to achieve operational goals, such as minimizing vehicle waiting time. Several studies demonstrated that dispatching rules have a significant impact on the performance of the system, such as the average transfer time, waiting time, vehicle utilization, and even throughput [15–18]. Some investigations introduced new dispatching rules by reassigning vehicles during the unloading travel time [19–21]. The last problem to improve AMHS operation is to design efficient vehicle routing rules. The vehicle routing problem determines the optimal vehicle routes to visit, thereby aiming to minimize transport times of TRs. Typically, studies on vehicle routing problems have focused on identifying conflict-free routes [22–25]. Another objective is to design vehicle routing decisions that consider traffic congestion, which should be avoided insofar as possible [26, 27].

More recently, studies on improving AMHS have expanded their scope to include storage or lifter assignment. This expanded scope is inevitable owing to the fact that many FABs are starting to use multiple lines (floors) to increase their production capacity. Kim et al. [28] and Siebert et al. [29] focused on lot targeting to improve the material flows through the storage locations, while Lee et al. [30] proposed a machine learning approach to select the best dispatching rule for storage allocation. They showed that the storage allocation significantly affects the performance of the AMHS.

Although the lifter assignment problem has emerged as an important issue in the field of semiconductor manufacturing, to the best of the authors’ knowledge, only three published works studied this problem [1, 31, 32]. The first study to introduce the lifter assignment problem for inter-floor transportation is by Jimenez et al. [31]. They suggested four rules...
for the selection of the rail for TRs to be sent to a stocker on
the same floor, and four rules for the selection of the lifter
to be sent to another floor. They recommended considering
both travel time and the number of waiting lots when the
number of inter-line transportation changes. Na et al. [1]
proposed the shortest-expected-arrival-time (SEAT) rule to
select a lifter and proved that this rule outperforms the
methods used in practice. Lee et al. [32] proposed operation poli-
cies to improve the efficiency of lifter operations in mate-
rial handling of semiconductor lines. Their policies involve
the specific and practical operational decisions, such as the
number of virtual ports, activating the alternative storage,
and using a shelf extraction procedure by rack-masters. How-
ever, they did not consider the nature of stochastic dynamics
which is considered the inherent characteristic of the lifter
assignment problem.

Based on the literature review above, we believe that our
study contributes to it in three ways. First, our optimiza-
tion model is the first to consider stochastic dynamics in the
lifter-assignment problem. Our proposed model considers
not only the current state but also the states that will occur
in the future. Second, we propose an algorithm that can be
applied in a real-world setting. In general, MDP models have
difficulty finding an optimal solution when the target system
is large or complex. We address this limitation by clustering
the source machines with similar travel-time distributions
and by partitioning the problem into sub-problems based on
lifter groups. Lastly, we propose a framework with autono-
mous control that can serve as the basis for establishing a
smart factory. Since operators adjust the rules empirically
by the rack-master. The decision to assign the lifter should
determine which lifter to send the TR to at the moment the
TR is generated. Therefore, we developed an MDP model
that aims to minimize the expected delivery time. The
components of the proposed MDP model are now herein
detailed.

3 Mathematical model

We consider minimization of the inter-floor transportation
time as the lifter assignment problem. We only consider the
travel time on the source floor and waiting time at the lifter
because these are affected directly by the lifter assignment.
To consider the trade-off between the travel time on the
source floor and waiting time at the lifter in a probabilistic
manner, we mainly formulated this problem using MDP.

MDP is a modelling technique that takes into account the
immediate rewards (or costs) of current decisions and their
effect on the entire system under uncertainty [33].

An MDP model consists of system state \( s \), decision \( a \),
cost \( (C(s,a,s')) \), and state transition probability \( (p(s,a,s')) \),
where \( s' \) is the next state. The system is in one of the possible
states in the state set \( S \). At each decision epoch, the decision
maker observes the current state \( s \) of the system and chooses
an action \( a \) from a set of possible actions \( A(s) \). When
the selected action is performed, the current system state
changes to a new state \( s' \) according to the transition prob-
ability \( p(s,a,s') \). The decision maker receives a cost \( C(s,a,s') \)
when certain actions are performed, or certain conditions
are satisfied.

Using these quantities, the decision to minimize the value
function of the state is derived. The value function is formu-
lated using Bellman’s equation [33]:

\[
V(s) = \min_{a \in A(s)} \sum_{s'} p(s,a,s') \times \left( C(s,a,s') + \gamma V(s') \right)
\]

where \( \gamma \) is a discount factor satisfying \( 0 \leq \gamma < 1 \). Finally, a
policy that maps each state to the optimal action is derived.

The lifter assignment problem can be viewed as a sequen-
tial decision-making problem because it is necessary to
determine which lifter to send the TR to at the moment the
TR is generated. Therefore, we developed an MDP model
that aims to minimize the expected delivery time. The
components of the proposed MDP model are now herein
detailed.

System state In the proposed model, the system state is
defined as \( S = (TR_1, ..., TR_i, ..., TR_L, b_1, ..., b_j, ..., b_L) \), where
\( TR_j \) is the number of TRs transporting to lifter \( j \), \( b_j \) is the
number of TRs waiting in the buffer of lifter \( j \), and \( L \) is the
number of lifters. The number of lots that can be queued
in the lifter’s in-buffer is limited. When it exceeds the lifter
buffer capacity, an OHT carrying a wafer lot cannot load
into the buffer, and it circles around or waits on the road until
room becomes available. This not only increases OHT travel
time, but also creates congestion around the lifter. Therefore,
an upper bound \((UB)\) is set in advance on the sum of the
number of TRs moving to the lifter and the number of TRs
waiting in the lifter buffer. Therefore, the state set \( S \) of this
model consists of states where \( TR_j + b_j \) is equal to or smaller
than \( UB \) and greater than or equal to 0 for all \( j \).

Transition probability The system state changes to other
state when one of the following three events occurs: arrival
of a TR, loading the lot into the lifter, and picking up the lot
by the rack-master. The decision to assign the lifter should
be made when the arrival of a TR event occurs. An example
of a system with three lifters is shown in Fig. 3.

In Fig. 3, the system state in the centre of the figure is the
current state. The first three of the six numbers in the state
indicate the number of TRs moving to each lifter, and the last
three indicate the number of TRs waiting at an in-buffer port of
each lifter. The event of current state changing to the next state
is described in the dotted square box. The arrows indicate the
probability of occurring of each state transition. For example, when \( \text{Loading the lot into lifter 1} \) occurs, since the number of TRs moving to lifter 1 decreases by 1 and the number of waiting TRs in lifter 1 increases by 1, the current state \((1, 3, 1, 2, 1, 2, 1)\) changes to the next state \((1, 3, 1, 2, 2, 1)\). In particular, when \( \text{arrival of a TR} \) occurs, it is a decision epoch because a lifter should be selected to transfer the lot, and the next state is determined by the lifter assignment.

Based on the results of the preliminary analysis that fits historical data into some mathematical distributions, we assume that the inter-arrival times for all events follow an exponential distribution. Under this assumption, the event generation process of this model follows the Poisson process. Therefore, the system state transition probability is given by the arrival rate of each event divided by the sum of all rates \([33, 34]\), where \( \lambda_i \) denotes the inter-arrival rate of TR from source machine \(i\), \( TR_j \) denotes the number of TRs proceeding to lifter \(j\), \( \mu_j \) denotes the travel time to the lifter \(j\), and \( w_j \) represents the cycle time of the lifter \(j\). In Eq. (2), the first term is the rate at which TRs arrive from each of the \(M\) source machines, and the second term is the rate at which OHTs arrive at each lifter and load lots into them. The last term is the rate at which the lifter’s rack-master transports the lot to another floor and returns to the departure floor. The indicator function \(1_{\{b_j > 0\}}\) takes a value of 1 when there is at least one waiting TR in the lifter and a value of 0 when there is no waiting TR. Finally, the transition probabilities for the three events are as follows:

- **Arrival of a TR** (from the source machine \(i\))
  \[ \rho = \frac{\lambda_i}{\rho}, \quad i \in \{1, \ldots, M\}, \]
- **Loading the lot** (into the lifter \(j\))
  \[ \frac{TR_j \times \mu_j}{\rho}, \quad j \in \{1, \ldots, L\}, \]
- **Picking up the lot** by the rack-master (of the lifter \(j\))
  \[ \frac{1_{\{b_j > 0\}} \times w_j}{\rho}, \quad j \in \{1, \ldots, L\}. \]

**Decision** When a TR arrives, the system determines which lifter will be used for the inter-floor transportation. According to the state set definition, when the sum of the number of TRs proceeding to lifter \(j\) and the number of TRs waiting in the buffer of lifter \(j\) \((TR_j + b_j)\) reaches the predetermined
upper bound \((UB)\), the lifter \(j\) is not selected until the value decreases.

**Cost function** Two cost function types are used. When a TR is generated and a lifter is selected for the inter-floor transportation, the cost is defined as the average travel time from the source machine to the lifter. However, when the TR arrives at the lifter, the cost is the waiting time, which is calculated by multiplying the cycle time of the lifter by the number of jobs waiting in the buffer. Formally, the cost \(C\) in state \(s\) with decision \(a = j^*\) is denoted as

\[
C(s, a = j^*) = \begin{cases} \overline{TT}_{ij^*}, & \text{if a TR is generated from source machine } i \text{ and lifter } j^* \text{ is selected}, \\ C_{T_j} \times b_j, & \text{if the TR arrives at lifter } j^*, \\ 0, & \text{otherwise}, \end{cases}
\]

(3)

where \(\overline{TT}_{ij^*}\) is the average travel time from source \(i\) to lifter \(j^*\) and \(C_{T_j}\) is the average cycle time of lifter \(j^*\).

**Objective function** The goal of the decision problem is to minimize the expected delivery time, which is the sum of the travel time from the source machine to the lifter and the waiting time in the buffer. The objective function is accordingly formulated using Bellman’s equation:

\[
V(s) = \min_{j^* \in \{1, \ldots, L\}} \sum_{i=1}^{M} \frac{\lambda_i}{\rho} \times [C(s, j^*) + \gamma V(s + e_{j^*})] + \sum_{j=1}^{L} \frac{\mu_j}{\rho} \times [C(s, j) + \gamma V(s - e_j + e_{M+j})] + \sum_{j=1}^{L} \frac{1}{\rho} \times [C(s, j) + \gamma V(s - e_{M+j})], \forall s \in S
\]

\[
s.t. s + e_j \in S, \forall s, j^* \\
s - e_j + e_{M+j} \in S, \forall s, j \\
s - e_{M+j} \in S, \forall s, j
\]

(5)

(6)

(7)

We define \(e_k\), a \(1 \times |2M|\) unit vector as the change in the value of \(k\text{th}\) position. Equation (5) constrains the state transition to the next state by the selection of lifter \(j^*\) after an Arrival of a TR event. Constraint (6) means the decrement of \(TR\) and the increment of \(b_j\) by loading the lot event. The decrement of waiting lots in the lifter buffer is described in Eq. (7) by picking up the lot by the rack-master event. That is, a decision cannot be made in which the next state is not included in the predefined state set \((S)\).

### 4 Solution method for lifter assignment in an actual fab

Although the MDP model in Sect. 3 guarantees the provision of an optimal policy for decisions on a lifter assignment, there exists a critical computational issue from a practical standpoint. An MDP model is generally solved by using dynamic programming (DP). However, when the problem size approaches the size typically found in real-world applications, the MDP model requires tremendous computing resources and time to handle the large number of variables. Because of this “curse of dimensionality”, solutions of realistic problems usually cannot be obtained in a reasonable amount of time. Considering the size of an actual FAB and its complex operating environment, the size of the state space must be very large. To tackle this difficulty, we propose a solution method to efficiently solve the given problem.

#### 4.1 Solution approach

The main idea of the proposed solution approach is to reduce the dimension of the original problem while keeping the quality of its solution. The dimension of the problem is determined by the number of source machines and destination lifters. In our approach, we employ the concepts of clustering, partitioning, and tournament. First, the dimension of the source machines is reduced by grouping numerous machines into similar clusters. Second, we create several sub-problems by partitioning the destination lifters into sub-groups of manageable size. Then, the optimal policies are obtained for the sub-problems and all possible scenarios derived during the tournament. Finally, in the execution phase, the best lifter for each TR is determined by the tournament among the sub-problems.

Modelling each machine individually increases the complexity of the MDP model significantly because a typical FAB usually has hundreds to thousands of machines. Thus, we cluster the machines based on similar travel times from the machines to the lifters. For example, 40 machines are grouped into 7 clusters as shown by different colours in Fig. 4. The machines within a cluster have similar distributions of travel time to destination lifter.

Next, by partitioning the lifters into different groups, we divide the problem into several sub-problems that can be solved optimally within a reasonable computational time. Thus, the number of lifters in each sub-problem is limited to a maximum of \(N\) lifters. For instance, as in Fig. 5, the original problem is divided into 2 sub-problems coloured in red and blue, with a maximum of three lifters each. This
means that the dimensionality of the original problem is cut by the number of sub-problems.

The next step is to obtain the optimal policies for each sub-problem. In order to obtain the final best lifter, it is also necessary to solve new sub-problems consisting of the optimal lifters selected from the sub-problems. That is, all optimal policies for all possible problems have to be prepared a priori. In Fig. 6, the nine scenarios that are possible based on a combination of two sub-problems that have three results, respectively, are shown. It is worth noting that if the number of groups is greater than \( N \), the selected lifters are regrouped in the manner described above and the tournament selection method is applied.

The final step is to select the best lifter for each TR in the execution phase within a short time, almost real-time. When a new TR is issued, the best lifter is derived by the tournament selection that only selects the optimal policies, which have already been solved in the previous step 3 (Fig. 7).

### 4.2 Algorithms for lifter assignment

The detailed procedure is presented as pseudo-code in Tables 1 and 2. Considering the real operation of FABs, a lifter assignment algorithm using MDP is not necessarily updated in a real-time manner owing to the fact that the optimal policies can be prepared by considering all possible situations a priori. However, a lifter assignment step should be activated in real time whenever a TR for inter-floor transportation arrives. By considering this circumstance, we devised two algorithms for the actual FAB operations. One is the **Policy Preparation Module**, and...
which constructs MDP models and derives lifter assignment policies using historical data periodically; the other is the Assignment Module, which assigns lifters in real time using policies derived in the Policy Preparation Module.

Table 1 presents the pseudo-code for the Policy Preparation Module. In step 1, using the historical data of which machines sent TRs to all lifters, the source machines with similar travel time distributions for each lifter are grouped together. In our case, the average travel time from the machine to each lifter is expressed as a vector. Machines that are close to each other are assigned to the same cluster, and in the case of a machine with no machines in close proximity, a new cluster for that machine is then created. The details of the algorithm and parameters used in our study are presented in Sect. 5.1.

Next, in order to assign cluster IDs for machines that did not send TRs to all lifters, we create a classifier, \( f(x, y) \), by learning the \( x \) and \( y \) coordinates and cluster IDs. By using this classifier, all travel time clusters can be grouped into clusters. After allocating a cluster to all machines, the average travel time from each cluster to each lifter (\( \bar{T}_{jk} \)) and the inter-arrival time of the TR in each cluster (\( \bar{T}_{kj} \)) are obtained using data of the machines in each cluster.

In a similar way, in step 2, we group the lifters using their physical location information, \( IG = \{ IG_1, \ldots, IG_L \} \). That is, by using the coordinates \( x \) and \( y \) of all lifters, we cluster up to \( N \) lifters as one group. \( N \) can be set according to the computing environment in which the algorithm is used to manage the computational load. Then, we form additional groups, \( AG \), for all combinations that can be created when one lifter is picked from each group. For example, we create initial lifter groups for seven lifters with \( N \) set to three using the coordinates \( x \) and \( y \) of all lifters, \( IG = \{ (1,2,3), (4,5), (6,7) \} \). Based on these initial groups, the following additional groups are created: \( AG = \{ (1,4,6), (1,4,7), (1,5,6), (1,5,7), (2,4,6), (2,4,7), (2,5,6), (2,5,7), (2,4,6), (2,4,7), (2,5,6), (2,5,7) \} \).

In step 3, we derive the optimal policy for all lifter groups formed in step 2 (\( \pi_{IG}, \forall g \in \{ 1, \ldots, L \} \) and \( \pi_{ag}, \forall ag \in AG \)). In other words, given the current state and source machine, a policy that determines the best lifter among lifters in the group is obtained. Optimal policies of additional lifter groups, \( \pi_{ag}, \forall ag \in AG \), are used when selecting the final optimal lifter through the tournament method in the Assignment Module. The best lifter in each initial group is determined by the current system state and cluster ID of the source machine. Since we do not derive the optimal policy from the model constructed in real time, we must consider all combinations of lifters that can be selected from each initial group. Therefore, by selecting one lifter from each group, we create additional groups for all cases that can be made and obtain the optimal policy of these groups.
Table 2 shows the pseudo-code for the Assignment Module. When a TR is generated from source machine \( i \), the Assignment Module checks the current state \((S)\) and obtains the cluster ID of the source machine \((CID_i)\) using the classifier (Step 4-2 in the Assignment Module). The module obtains the best lifters \( \left( j_{IGg}, \forall g \in \{1, \ldots, L_N\} \right) \) from the optimal policy of each initial lifter group \( \left( \pi_{IGg}(S, CID_i) \right) \) by using the machine’s cluster ID and current state as inputs (Step 4-3 in the Assignment Module). Finally, in Step 4-4, the module finds a lifter group \( \left( ag = \left\{ j_{IG1}, \ldots, j_{ILN} \right\} \right) \) consisting of lifters selected from each initial group and selects the optimal lifter through the optimal policy of the group \( \left( \pi_{ag}(S, CID_i) \right) \). Let us consider the previous example again. If lifter 1, lifter 4, and lifter 7 are selected as the optimal lifters in each initial lifter group, \( IG=\{ (1,2,3), (4,5), (6,7) \} \), we use the optimal policy of the group \( (1, 4, 7) \), \( \pi_{(1,4,7)} \).

### 4.3 A framework for system implementation

The proposed solution and algorithms theoretically enable effective lifter assignment in actual FABs. Figure 8 shows the system framework to apply these algorithms to real FAB operation. The two algorithm modules are implemented on two system modules of the same names.

The Policy Preparation Module produces optimal policies for lifter groups and cluster information for source machines by periodically using historical data. All these outputs are stored in the central system of AMHS in table format. The table of the optimal policy contains the system state, cluster ID, lifter group ID, and the optimal lifter ID in that group as columns. Using this table, given the current state and cluster ID, the best lifter in each group can be selected easily. The cluster information table stores the results from mapping the machine ID to the cluster ID.

In the Assignment Module, when a TR for inter-floor transportation is generated, the state manager checks the current system state (the number of TRs moving to each lifter and the number of TRs waiting to be transported at each lifter). Then, the decision manager uses as input the name of the source machine that sent the TR and the current state. By automatically processing the information, it obtains the \( x \) and \( y \) coordinates of the source machine and reads its cluster ID from the cluster information table. Next, using the current state and cluster ID, the decision manager identifies the best lifter for each initial group and
determines the best lifter by the tournament. Thereafter, the wafer lot is transported to the lifter determined in the Assignment Module, and the transport records are stored in the history data table. These stored data are periodically used by the Policy Preparation Module.

5 Experimental results

To verify the effectiveness of the proposed approach, we conduct both a simple analysis in a virtual environment and a simulation study that mimics a real-world FAB using an AMHS-oriented simulation model (emulator) offered by that company. Since it describes the actual FAB operating environment and is guaranteed to be a high-fidelity simulation, it is an appropriate testbed to examine various approaches to designing effective operational strategies for FABs.

We use three benchmarking rules: the round robin rule, the shortest-transfer-time rule used in the actual FAB, and the SEAT rule which showed the best performance in a recent study [1]. The round-robin rule distributes TRs evenly to all lifters. It primarily aims to balance utilization of all lifters. Since the distance from the source machine to the lifter is not considered, it is not expected to reduce

| Table 1 Pseudo-code for the operation of a real FAB — Policy Preparation Module |
|-------------------------------------------------|
| **Input:** | M = \{i \in \{1, \ldots, M\}\} : machine ID set |
| | L = \{j \in \{1, \ldots, L\}\} : lifter ID set |
| | T{T}_{ij} : the travel time from source machine i to lifter j, \forall i \in M, \forall j \in L |
| | C{T}_{j} : the cycle time of lifter j, \forall j \in L |
| | \tau_i : the inter-arrival time of TRs in machine i, \forall i \in M |
| | \langle x, y \rangle_i : x, y coordinates of machine i, \forall i \in M |
| | \langle x, y \rangle_j : x, y coordinates of lifter j, \forall j \in L |
| **Step 1 Cluster source machines** |
| Step 1-1 Make a dataset for clustering, \( D = \{i \in M | T{T}_{ij} > 0, \forall j \in L\}\) (machines which sent TRs to all lifters) |
| Step 1-2 Get clustering result, \( \{(i, CID_i), \forall i \in D\} \leftarrow ClusterAlgorithm(D) \) (CID_i : cluster ID of machine i) |
| Step 1-3 Create a classifier which assigns a cluster ID for a machine, CID \leftarrow f(x, y), by learning \( \{(x, y)_i, CID_i), \forall i \in D\} \) |
| Step 1-4 Calculate \( T{T}_{kj}, \tau_i, C{T}_{j} \forall k \in \{1, \ldots, N_{CID}\}, \forall j \in L \) |
| **Step 2 Divide into sub-problems** |
| Step 2-1 Make initial lifter groups \( IG = \{IG_1, \ldots, IG_{\lfloor \frac{L}{N} \rfloor}\} \) using \( \langle x, y \rangle_j, \forall j \in L \left(2 \leq n(IG_g) \leq N, \forall g \in \{1, \ldots, \lfloor \frac{L}{N} \rfloor\}\right) \) |
| Step 2-2 Make additional groups, \( AG \), for all combinations that can be made when one lifter is picked from each group |
| **Step 3 Prepare the optimal policies for all possible scenarios a priori** |
| Derive the optimal policy, \( \pi_{IG_g}, \) which determine one among the lifters in \( IG_g, \forall g \in \{1, \ldots, \lfloor \frac{L}{N} \rfloor\} \) |
| Derive the optimal policy, \( \pi_{ag}, \) which determine one among the lifters in each additional group \( \{ag\}, \forall ag \in AG \) |
| **Output:** \( \pi_{IG_g}, \forall g \in \{1, \ldots, \lfloor \frac{L}{N} \rfloor\} \) : the optimal policy for each initial lifter group |
| \( \pi_{ag}, \forall ag \in AG \) : the optimal policy for each additional lifter group |
| \( f(x, y) \) : a classifier which assigns a cluster ID for a machine using \( x, y \) coordinates of a machine |
The shortest-transfer-time rule aims to transfer the TR to another floor as quickly as possible by selecting a lifter for which the weighted sum of the travel time between the two machines and the number of TRs being transferred is the smallest. In practice, the weight may be adjusted by operations managers depending on changes in the operating environment. In our experiment, the weight is set from the historical data. Finally, the SEAT rule assigns the lifter which is expected to have the shortest time for the lot to reach the rack-master. Mathematically, this rule can be expressed as:

\[
\min_j \sum_{l \in \{1, \ldots, L\}} \text{Current scheduled TR count of Lifter}_j \times \text{Cycle time of Lifter}_j + \text{Expected travel time of Lifter}_j + \text{Cycle time of Lifter}_j
\]

As explained in Sect. 4, the proposed algorithm uses clustering techniques. We cluster source machines using the well-known DBSCAN clustering algorithm [35]. DBSCAN assigns a cluster ID if there are at least \(\text{minPts}\) data points within the distance \(\epsilon\) from one data point \(p\). The algorithm parameters \(\epsilon\) and \(\text{minPts}\) are set to 0.2 and 3, respectively, and the Euclidean distance is used for the distance measure. To train faster and reduce the likelihood of falling into local

---

**Table 2** Pseudo code for the operation of a real FAB — Assignment Module

| Input:  | \(i\) : Source machine ID |
|---|---|
| **Step 4** Obtain the best lifter by the tournament selection |
| Step 4-1 | Get the current state, \(S = (TR_1, \ldots, TR_{L}, b_1, \ldots, b_L)\) |
| Step 4-2 | Get the cluster ID of \(i\), \(CID_i \leftarrow f((x, y)_i)\) |
| Step 4-3 | Get the best lifter among lifters in each initial lifter group given \(S\) and \(CID_i\) |
| \(j_{ig}^* \leftarrow \pi_{ig}(S, CID_i), \forall g \in \{1, \ldots, \left\lfloor \frac{L}{N} \right\rfloor\} \) |
| Step 4-4 | Get the optimal lifter from the optimal policy of \(ag\) consisting of \(j_{ig}^*, \forall g \in \{1, \ldots, \left\lfloor \frac{L}{N} \right\rfloor\} \) given \(S\) and \(CID_i\) |
| \(j^* \leftarrow \pi_{ag}(S, CID_i), \quad ag = \{j_{i1}^*, \ldots, j_{i\left\lfloor \frac{L}{N} \right\rfloor}^*\} \) |
| **Output:** | \(j^*\) : the optimal lifter to be sent TR |

---

Fig. 8 System implementation framework for lifter assignment in real FABs
optimal states, we standardize the average travel time from the machine to each lifter used as the data point. In addition, we individually assign cluster IDs to source machines that are judged as noise points because they do not belong to any cluster. To create the classifier, the \( k \)-nearest neighbour (\( k \)-NN) algorithm is used. We set the parameter \( k \) of the \( k \)-NN algorithm to 1 so that the cluster closest to the input machine is assigned. The Euclidean distance is also used as a distance measure in the \( k \)-NN algorithm. Finally, we derive the optimal solution of the MDP model by using the value iteration algorithm. The discount factor \( \gamma \) of the MDP model is set to 0.99, and the stopping criterion required for the value iteration algorithm is set to 0.001.

The proposed algorithm is implemented in the Java programming language. The computing environment used in the experiment is as follows: Intel Xeon 6146 (3.20 GHz), 16 GB RAM, Windows 10 Enterprise.

### 5.1 Experiments for the simple virtual problem

We first analyse its performance with a small-sized testbed. There are three lifters and eight machines. The travel times for all combinations of lifters and machines are shown in Table 3. The inter-arrival times for each machine, assumed to be random, are also shown. The values in the Table 3 are the mean values of the exponential distribution. The cells coloured grey indicate the shortest travel times from each source machine to the lifter.

Under the conditions shown in Table 3, optimized policy for assigning lifters is derived by utilizing the MDP model proposed in Sect. 3. Table 4 shows two notable cases from the optimized policy. First, since the objective of the MDP model is to minimize the total of the travel and waiting times, the lifter with the smallest value of \( TR_j + b_j \times \text{weight} \) is selected. For instance, in the case of source machine 4, lifter 2 is selected by comparing the values of \( TR_j + b_j \times \text{weight} \) for each lifter. (Unlike the shortest-transfer-time rule, the weight is optimally obtained from the model by considering the given system environment variable and future state.)

Secondly, case 2 reveals that the proposed approach can consider future system states when deriving an optimized policy. Assuming that all values comprising the state are zero (i.e., there are no TRs), source machines 1, 2, 3, 4, and 8 should all select lifter 1 because the travel time to it is the shortest. However, source machine 4 selects lifter 2 instead. This is because lifter 1 is frequently called by other source machines (1, 2, 3, and 8). This may generate severe congestion by making the lifter-assignment policy sub-optimal. To avoid such inefficiency, the proposed approach selects another lifter, thereby distributing the traffic volume and balancing the load. Thus, the lifter-assignment policy derived from the proposed approach can consider not only the immediate travel time but also the subsequent scenarios.

### 5.2 Experiments for a real problem

We construct a simulation study based on historical data gathered from Samsung Semiconductor. The FAB we used for our simulation study consists of more than 800 source machines, more than 150 OHTs, and 7 lifters. We use data from March 2020; more detailed information about the data cannot be disclosed because of security issues.

#### 5.2.1 Clustering results

Figure 9 shows the clustering results for the actual FAB. The symbol X coloured in black represents the location of

| Source machine ID | Lifter ID | Inter-arrival time |
|-------------------|-----------|--------------------|
| 1                 | 50        | 70                 |
| 2                 | 60        | 90                 |
| 3                 | 55        | 70                 |
| 4                 | 100       | 110                |
| 5                 | 125       | 105                |
| 6                 | 100       | 70                 |
| 7                 | 85        | 115                |
| 8                 | 55        | 120                |

| Cycle time |
|------------|
| 35         |
| 40         |
| 30         |
the lifter; a circle indicates each source machine’s location. Circles of the same colour refer to the same cluster. Forty clusters are formed in the FAB used in the experiment.

The clustering outcomes shown in Fig. 9 appear generally reasonable: machines located near a lifter are grouped together because of their relatively short travel times, whereas machines that are located far from the lifters and have long travel times are similarly clustered.

5.2.2 Performance evaluation with emulator

In order to evaluate the performance of the four lifter assignment rules (including the three benchmarking rules), three performance indicators are measured. For TRs sent to lifters, the average travel time, the average waiting time, and the number of completed TRs are recorded (a warm-up period of 30 min is set to reach the steady state of the model).

Table 5 shows the summary of results from the four assignment rules. The proposed algorithm outperforms the benchmarking rules in all three measures. For TRs sent to lifters, when the proposed algorithm is implemented, the average travel time is reduced by 19.7% (165.2 s → 132.6 s) over that of the round-robin rule, 7% (142.7 s → 132.6 s) over that of the shortest-transfer-time rule and 5.4% (140.2 s → 133.6 s) over that of SEAT rule. The round robin rule showed the best result for average waiting time as it intends to distribute the workload evenly to lifters. However, the total time was showed worst performance because the increase in travel time was much greater than the reduction in waiting time. The shortest-transfer-time rule showed the worst performance in waiting time because it does not consider the waiting time at lifters. The SEAT rule was generally superior to the two rules commonly used in an actual FAB because it considers both travel time and waiting time. Our algorithm showed the shortest sum of the average travel time and the average waiting time, although the waiting time was slightly longer than that of the round-robin rule. Our algorithm always performed better than SEAT rule.

| Lifter assignment rule | Round robin | Shortest transfer time | SEAT | Proposed algorithm |
|------------------------|-------------|------------------------|------|-------------------|
| Number of completed TRs| 2327        | 2336                   | 2347 | 2348              |
| Average travel time (s)| 165.2       | 142.7                  | 140.2| 132.6             |
| Average waiting time (s)| 15.6        | 29.6                   | 20.3 | 18.5              |
| Total time (s)         | 180.8       | 172.3                  | 160.5| 151.1             |
This is because our algorithm considers even the possible future states of the system, whereas the SEAT rule only considers the current states.

Another notable finding, which may be observed in the graphs in Fig. 10, is that the proposed algorithm also yields the shortest travel time for most of the time windows. One unit on the x axis in Fig. 10 is 10 min. It is worth noting that the proposed algorithm showed fewer TRs with a travel time of more than 140 s, even though it handled slightly more TRs than the other rules. These results indicate that the proposed algorithm prevents delivery delays. Although the shortest-transfer-time rule assigns wafer lots to the nearest lifters, travel time was not the best because TRs was concentrated on particular lifters, creating congestion around the lifters.

The economic implications of the above results are that fewer OHTs will be required than in the previous scenario. Since the load factor of an OHT is a function of its travel time, we can calculate the OHT-investment cost savings. Based on the simulation results shown above, we estimate that approximately 2.5 million USD (approximately 2.76 billion KRW) could be saved if the proposed algorithms are implemented in actual FABs operating approximately 3000 OHTs. In addition, the decrease of transfer delay is another expected benefit.

5.2.3 Sensitivity analysis

To check the robustness of the proposed algorithm, we conduct a sensitivity analysis to help predict its performance under unusual situations in which the load factors of OHTs and lifters increase in accordance with TRs. To examine such situations, we arbitrarily increase the arrival rate of the next TR to be 1.05 times, 1.1 times, 1.15 times, or 1.2 times higher than the historical data.

The results of the sensitivity analysis are shown in Table 6. Of the three measures, we illustrate as representative the average travel time, which is regarded as the most important measure in practice. Table 6 shows that if the arrival rate of the next TR increases, the travel time to lifters also increases regardless of the type of assignment rule. Nonetheless, the proposed algorithm has the shortest travel time, even for the highest arrival rates we consider. To confirm this statistically, we conduct a t-test to examine

| Lifter assignment rule       | Change of arrival rate |
|------------------------------|------------------------|
|                              | ×1.00 | ×1.05 | ×1.10 | ×1.15 | ×1.20 |
| Proposed                     | 132.5 | 136.6 | 139.4 | 143.1 | 146.1 |
| SEAT                         | 140.2 (<0.05) | 142.8 (<0.05) | 149.0 (<0.05) | 153.1 (<0.05) | 155.0 (<0.05) |
| Shortest transfer time       | 142.7 (<0.05) | 147.5 (<0.05) | 150.7 (<0.05) | 158.7 (<0.05) | 159.3 (<0.05) |
| Round robin                  | 165.2 (<0.05) | 168.2 (<0.05) | 172.5 (<0.05) | 174.7 (<0.05) | 178.0 (<0.05) |
the difference among the results. Based on the t-tests with a p-value <0.05, we conclude that the improvement in performance owing to the proposed algorithm is statistically significant.

Next, we test the performance of the proposed method by using out-of-sample data (specifically, a different dataset). As shown in Table 7, the travel times are increased for the differences among the results. Based on the t-tests with a p-value <0.05, we conclude that the improvement in performance owing to the proposed algorithm is statistically significant.

Next, we test the performance of the proposed method by using out-of-sample data (specifically, a different dataset). As shown in Table 7, the travel times are increased for all test instances when we use a different dataset. This is expected because the dataset used for deriving the optimal policy is different from that used for testing the performance. Nevertheless, it is worth noting that there is no statistically significant difference in three out of the five test instances. This suggests that even when the operating environment is altered by a small amount, the operating policy derived from the proposed approach will continue to perform adequately. This means that real-time updating of the lifter assignment policy, which requires tremendous computational resources, is not necessary. Hence, our approach of periodically updating the lifter assignment policy is valid in practice.

This series of experiments, both idealized and realistic, verified the effectiveness of the proposed approach. In particular, there is a significant improvement over the round-robin rule, which has no mechanism to consider travel time when TRs need transportation to a lifter. Moreover, our approach outperforms the fastest transfer option that does not consider a long-term perspective, showing the importance of accounting for a system’s future state when assigning TRs to lifters.

6 Conclusion

This study focuses on the inter-floor transportation problem in semiconductor FABs with multiple floors. We consider the stochastic dynamics of the lifter-assignment problem and propose the MDP model that adopts clustering, partitioning, tournament techniques to reduce complexity to manageable sizes. The simulation results demonstrated that our approach outperforms the three benchmark rules, reducing transportation time by up to 19.7% for a real problem.

Our approach can be applied to other similar problems, such as vehicle and job dispatching problems, and is expected to demonstrate more improved results than heuristic rules or deterministic models in stochastic situations. Moreover, when sufficient data are available, the MDP model can be easily utilized in a reinforcement learning model. Therefore, we expect the proposed model to have significant meaning in the digital twin environment where both physical and virtual twins generate large amounts of data. Finally, our study shows the possibility of autonomous control that can serve as the basis for establishing a smart factory. That is, instead of a manual work which is labour-intensive and sometimes results in significant variation in operational efficiency, our proposed framework enables autonomous control by automatically updating the model.

Funding This was supported by the Korea National University of Transportation in 2021. This research was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (No. NRF-2019R1F1A106365).

Declarations

Ethical approval Not applicable.
Consent to participate Not applicable.
Consent to publications Not applicable.
Conflict of interest The authors declare no competing interests.

References

1. Na B, Woo JE, Lee J (2016) Lifter assignment problem for inter-line transfers in semiconductor manufacturing facilities. Int J Adv Manuf Technol 86:1615–26. https://doi.org/10.1007/s00170-015-8327-0
2. Kim J, Yu G, Jang YJ (2016) Semiconductor FAB layout design analysis with 300-mm FAB Data: Is minimum distance-based layout design best for semiconductor FAB design? Comput Ind Eng 99:330–346. https://doi.org/10.1016/j.cie.2016.02.012
3. Agrawal GK, Heragu SS (2006) A survey of automated material handling systems in 300-mm Semiconductor FABs. IEEE Trans Semicond Manuf 19:112–20. https://doi.org/10.1109/TSM.2005.863217
4. Le-Anh T, De Koster MBM (2006) A review of design and control of automated guided vehicle systems. Eur J Oper Res 171:1–23. https://doi.org/10.1016/j.ejor.2005.01.036
5. Qiu L, Hsu W, Huang S, Wang H (2002) Scheduling and routing algorithms for AGVs: a survey. Int J Prod Res 40:745–60. https://doi.org/10.1080/00207540110091712
6. Xie C, Allen TT (2015) Simulation and experimental design methods for job shop scheduling with material handling: a survey. Int J Adv Manuf Technol 80:233–43. https://doi.org/10.1007/s00170-015-6981-x
7. Huang CJ, Chang KH, Lin JT (2012) Optimal vehicle allocation for an automated materials handling system using simulation optimization. Int J Prod Res 50:5734–46. https://doi.org/10.1080/00207543.2011.622311
8. Liao DY, Fu HS (2004) Speed delivery: dynamic OHT allocation and dispatching in large-scale 300-mm AMHS management. IEEE Robot Autom Mag 22:32. https://doi.org/10.1109/MRA.2004.1337824
9. Lin JT, Wu CH, Huang CW (2013) Dynamic vehicle allocation control for automated material handling system in semiconductor manufacturing. Comput Oper Res 40:2329–39. https://doi.org/10.1016/j.cor.2013.04.007
10. Wang FK, Lin JT (2004) Performance evaluation of an automated material handling system for a wafer fab. Robot Comput Integr Manuf 20:91–100. https://doi.org/10.1016/j.rcim.2003.08.002
11. Kim BI, Park J (2009) Idle vehicle circulation policies in a semiconductor fab. J Intell Manuf 20:709. https://doi.org/10.1007/s10845-008-0159-4
12. Vahdani B (2014) Vehicle positioning in cell manufacturing systems via robust optimization. Appl Soft Comput 24:78–85. https://doi.org/10.1016/j.asoc.2014.07.001
13. Lee S, Lim DE, Kang Y, Kim HJ (2021) Clustered multi-task sequence-to-sequence learning for autonomous vehicle repositioning. IEEE Access 9:14504–14515. https://doi.org/10.1109/ACCESS.2021.3051763
14. Schmaler R, Schmidt T, Schoeps M, Luebke J, Hupfer R, Schlaus N (2017) Simulation based evaluation of different empty vehicle management strategies with considering future transport jobs. In: Proceedings of the 2017 Winter Simulation Conference (WSC), vol 294. pp 1–12. https://doi.org/10.1109/WSC.2017.8248071
15. de Koster RBM, Le-Ahn T, Van der Meer JR (2004) Testing and classifying vehicle dispatching rules in three real-world settings. J Oper Manag 22:369–86. https://doi.org/10.1016/j.jom.2004.05.006
16. Egbelu PJ, Tanchoco JMA (1984) Characterization of automatic guided vehicle dispatching rules. Int J Prod Res 22:359–74. https://doi.org/10.1080/00207548408942459
17. Im K, Kim K, Lee S (2009) Effective vehicle dispatching method minimizing the blocking and delivery times in automatic material handling systems of 300 mm semiconductor fabrication. Int J Prod Res 47:3997–4011. https://doi.org/10.1080/00207540801914934
18. Kuo CH, Huang CS (2006) Dispatching of overhead hoist vehicles in a fab intrabay using a multimission-oriented controller. Int J Adv Manuf Technol 27:824–32
19. Bozer YA, Yen C (1996) Intelligent dispatching rules for trip-based material handling systems. J Manuf Syst 15(4):226–239. https://doi.org/10.1016/0278-6125(96)84549-3
20. Kim BI, Oh S, Shin J, Jun M (2007) Effectiveness of vehicle reassignment in a large-scale overhead hoist transport system. Int J Prod Res 45:789–802. https://doi.org/10.1080/00207540600675819
21. Le-Ahn T, De Koster MBM (2007) On-line dispatching rules for vehicle-based internal transport systems. Int J Prod Res 43:1711–28. https://doi.org/10.1080/00207540412331320481
22. Chen HK, Hseuh CF, Chang MS (2006) The real-time time-dependent vehicle routing problem. Transp Res Part E Log Transp 42:383–408. https://doi.org/10.1016/j.trb.2005.01.003
23. Kim CW, Tanchoco JMA (1991) Conflict-free shortest-time bidirectional AGV routing. Int J Prod Res 29:2377–2391. https://doi.org/10.1080/00207549108948090
24. Smolic-Rocak N, Bogdan S, Kovacic Z, Petrovic T (2010) Time windows based dynamic routing in multi-AGV systems. IEEE Trans Autom Sci 7:151–155. https://doi.org/10.1109/TASE.2009.2016350
25. Taghboni-Dutta F, Tanchoco JMA (2007) Comparison of dynamic routing techniques for automated guided vehicle system. Int J Prod Res 33:2653–69. https://doi.org/10.1080/00207549508945352
26. Bartlett K, Lee J, Ahmed S, Nemhauser G, Sokol J, Na B (2014) Congestion-aware dynamic routing in automated material handling systems. Comput Ind Eng 70:176–182. https://doi.org/10.1016/j.cie.2014.02.002
27. Lau H, Woo SO (2008) An agent-based dynamic routing strategy for automated material handling systems. Int J Comput Integr Manuf 21:269–288. https://doi.org/10.1080/09511970701241624
28. Kim H, Lim DE, Lee S (2020) Deep learning-based dynamic scheduling for semiconductor manufacturing with high uncertainty of automated material handling system capability. IEEE Trans Semicond Manuf 33:13–22. https://doi.org/10.1109/TSM.2020.2965293
29. Siebert M, Bartlett K, Kim H, Ahmed S, Lee J, Nazzal D, Nemhauser G, Sokol J (2018) Lot targeting and lot dispatching decision policies for semiconductor manufacturing: optimization under uncertainty with simulation validation. Int J Prod Res 56:629–41. https://doi.org/10.1080/00207543.2017.1387679
30. Lee S, Kim Y, Kahng H, Lee SK, Chung S, Cheong T, Shin K, Park J, Kim SB (2020) Intelligent traffic control for autonomous vehicle systems based on machine learning. Expert Syst Appl 144:113074. https://doi.org/10.1016/j.eswa.2019.113074
31. Jimenez J, Kim B, Fowler J, Mackulak G, Choung YI (2002) Operational modeling and simulation of an inter-bay AMHS in semiconductor wafer fabrication. Proc Winter Simul Conf 2:1377–1382. https://doi.org/10.1109/WSC.2002.1166405
32. Lee J, Kim S, Na B (2018) Optimization of lifter operations for inter-line transfers in semiconductor manufacturing facilities. Int J Prod Eng Res Dev 8:167–178. https://doi.org/10.24247/ijmeerdaug201819
33. Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons. https://doi.org/10.1002/9780470316887
34. Ross SM (1996) Stochastic processes. Wiley, New York
35. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial datasets with noise. Proc Sec Int Conf Knowl Discov Data Min 94:226–231

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.