Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

SIMULATION BASED MULTI-OBJECTIVE FAB SCHEDULING BY USING REINFORCEMENT LEARNING

Won-Jun Lee
Byung-Hee Kim

VMS Solutions Co.,Ltd.
U-Tower Building A #2001
Sinsu Street, 767
Yongin, 16827, REPUBLIC OF KOREA

Keyhoon Ko

VMS Global, Inc.
1952 Gallows Rd STE 110
Vienna, VA 20120, USA

Hayong Shin

Department of Industrial and Systems engineering
Korea Advanced Institute of Science and Technology(KAIST)
291 Daehak-ro, Yuseong-gu
Daejeon, 305-701, REPUBLIC OF KOREA

ABSTRACT

Semiconductor manufacturing fab is one of the most sophisticated man-made system, consisting of hundreds of very expensive equipment connected by highly automated material handling system. Operation schedule has huge impact on the productivity of the fab. Obtaining efficient schedule for numerous equipment is a very complex problem, which cannot be solved by conventional optimization techniques. Hence, heuristic dispatching rules combined with fab simulation is often used for generating fab operation schedule. In this paper, we formulate the fab scheduling problem as a semi-Markov decision process and propose a reinforcement learning method used in conjunction with the fab simulator to obtain the (near-)optimal dispatching policy. Resulting schedule obtained by the proposed method shows better performance than heuristic rules whose parameters are tuned by human experts.

1 INTRODUCTION

To improve performances of fabs in various aspects, many dispatching rules such as FIFO (First In First Out), SPT (Shortest Processing Time) or EDD (Earliest Due Date) have been studied. It is known that a single dispatching rule improves a particular performance of fabs (Haupt 1989; Đurasević and Jakobović 2018), however, it is nearly impossible to improve multiple performances at the same time by applying a single dispatching rule. Therefore, in the real fab, operators use the weighted sum of multiple scores derived from well-known dispatching rules or their scheduling experience. (Li et al. 2012; Li et al. 2013) introduced adaptive dispatching rules that can be calculated in certain amount of numeric values. The operators select a job that has the highest score when a dispatching event occurs. Even though the schedule is sensitive to the weights of the dispatching scores (Lee et al. 2018), it is still adjusted by human intuition.

There are studies on scheduling problem in manufacturing systems using Reinforcement Learning (RL). (Ramirez-Hernandez and Fernandez 2007; Ramirez-Hernandez and Fernandez 2009; Chen et al. 2010). The study of Ramirez-Hernandez and Fernandez (2007) introduces a basic formulation of the re-entrant line control problem, and a RL approach using a simple approximation. And they applied the method to the Intel Mini-Fab benchmark model (Ramirez-Hernandez and Fernandez 2009). In the literature, the fab state

is represented by buffer level and the occurrence of failure or PM on equipment. They introduce a learning algorithm that is capable of finding an optimal policy among 5 dispatching rules when the state feature is given, while focusing on minimization of WIP and cycle time. Chen et al. (2010) build a fixed-sized state-action pair table. The table contains 10 states, which are classified by WIP ratios, and actions, which are represented by predefined weight sets of dispatching rules. In addition, each element, that is an action-value function, in the table is estimated to find the optimal dispatching policy. Similarly, Shahrabi et al. (2017) applied RL approaches to the job shop scheduling problem with a 20 x 8 sized state-action pair table.

Ramírez -Hernandez and Fernandez (2009) focused on which dispatching rule to select. Chen et al. (2010) and Shahrabi et al. (2017) focused on which weight set to select. In this research, we focus on which job to select. This research introduces a RL approach that satisfies various purposes of a virtual fab, which contains re-entrant lines while retaining the idea of real fab operators. The state features defined in this research are capable of describing fab states regardless of the size of a fab. In addition, the direction of learning can meet the objectives of fab with intuitive settings for reward or penalty values.

2 PROBLEM DISCRIPTION

2.1 Re-Entrant Line Model

In the real fab, there are hundreds of machines and each of which takes charge of different processes, and the products have to undergo hundreds of process steps before completed. As shown in the existing studies (Ramirez-Hernandez and Fernandez 2007; Ramirez-Hernandez and Fernandez 2009), a fab can be represented as a re-entrant line model (Kumar 1993). During the production, the products have to visit hundreds of process steps and there are many cases in which products have to visit the same process steps several times. (i.e. the products are re-routed to some of the steps that have been visited.)

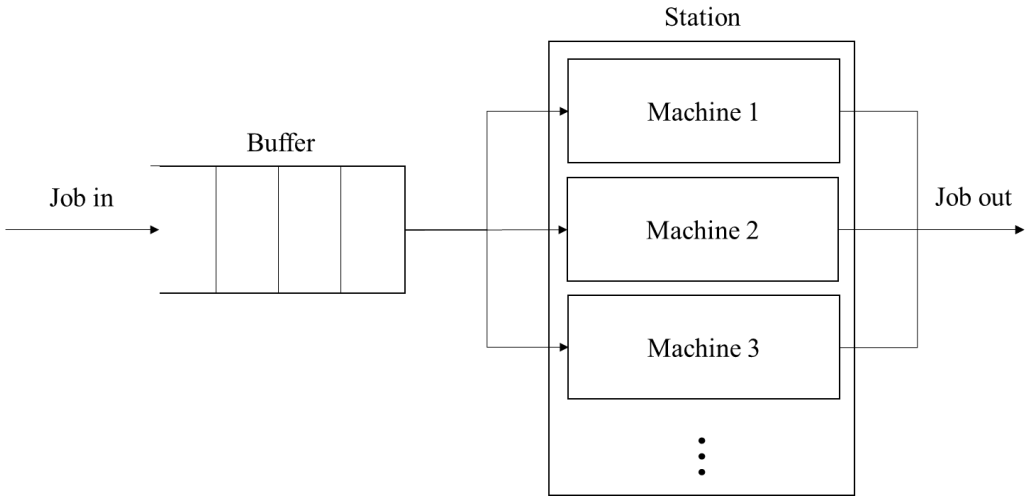

Figure 1: The fundamental unit of the fab line.

Figure 1 shows the fundamental unit of the fab line. A set of machines that execute the same process is called a station. Each station includes multiple machines and has buffers. A job arriving at the buffer waits until the suitable machines select it according to their dispatching logic. A relationship that indicates whether a job can be processed by a machine is called a job arrangement. Job arrangements vary on job types so that some of the jobs can be arranged to all machines in the station and some of them can be arranged to only one machine. If a machine loads a job has a different type from the last loaded job, a setup change occurs and takes some time for a tool change. A fab is a set of the fundamental units with complex connections based on process route of products as shown in Figure 2.

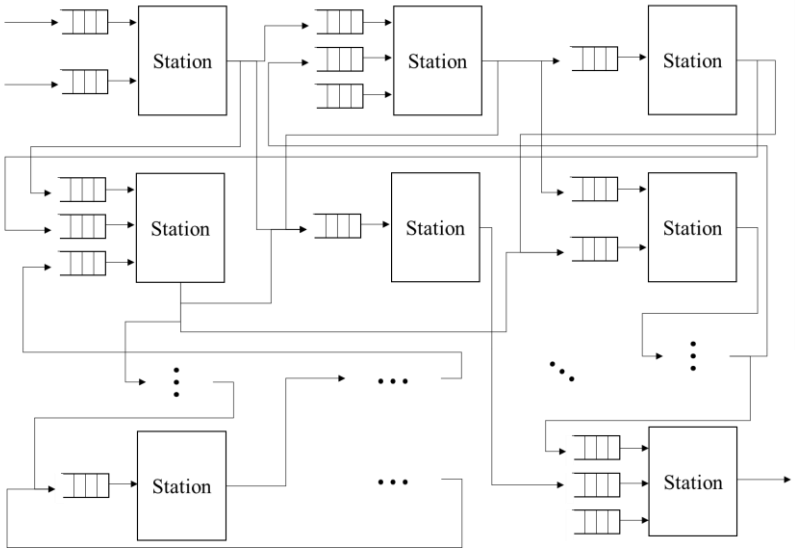

Figure 2 : General diagram for a re-entrant line fab

2.2 Dispatching Scores

As mentioned in the introduction, fab schedules are built by selecting the job that has the highest weight sum of scored dispatching rules. There are diverse dispatching scores by their purposes. Besides, some of well-known dispatching rules are transformed into scores and are used in fabs, and some of the fab operators create and use scores based on their needs and with concerns on their fab situations. The scores are including comprehensive information on the fab situation at the point of dispatching, and thus have high feature resolution in expression of a fab. This property is useful when applying RL and details will be described in Section 3.3

Table 1: Dispatching score examples.

Dispatching score	Description	Relate KPI
FIFO	Assign a large value to a job that arrives earlier than others.	
SPT/LPT	Assign a large value to a job that has shorter/longer processing time than others.	
Setup	Assign 0 if a job causes a setup change, otherwise 1.	Number of Setups
Required Eqp	Assign ratio of machines that have tool setting for a job to the entire machine in the current station.	Number of Setups
Queue Time	Assign a large value to a job close to the pre-defined queue time limit. If a job exceeds the limit, assign 0.	Queue Time Violation
Target Date	Assign a large value to a job close to the pre-defined due date. If a job exceeds the due date, assign 1.	Due Date Violation
Layer Change	Assign 0 if a job causes layer change, otherwise 1.	Number of Layer Changes

3 REINFORCEMENT LEARNING APPROACH

3.1 Markov Decision Process and Reinforcement Learning

The first step in the application of the RL is defining the problem to solve in the form of a Markov Decision Process (MDP). A MDP is specified by the collection of objects $< T, S, A, P(\cdot\,|s,a), R(s,a) >$ where T is a discrete time horizon, S is a state space, A is an action space, $P(\cdot\,|s,a)$ are the state transition probabilities and $R(s,a)$ is a reward function. Since fabs are complex systems consisting of various machines, jobs, processes, it is difficult to know state transition probabilities $P(\cdot\,|s,a)$ exactly. However, there exist some probability distributions which represent the system dynamics and if it is assumed that the system satisfies the Markov property in Equation (1), then it can be said that there is enough information needed in the application of the RL.

$$P(s_{t+1} = s'|s_t, a_t) = P\big(s_{t+1} = s'\big|s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, s_{t-3}, a_{t-3}, \cdots, s_0, a_0\big) \tag{1}$$

RL is a set of algorithms can handle MDP based control problem under large state, action spaces and uncertain state transition probabilities. Most of RL algorithms are based on the estimation of the value functions. The definition of value function is as follows:

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots |s_t = s] = E_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s],$$

$$Q^{\pi}(s,a) = E_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s, a_t = a].$$

The value functions are functions of states or state-action pair which represent the expectation of the sum of discounted future rewards under π which is the policy of decision makers. By estimating a value function under policy π, we can get better policy π' according to following equation:

$$\pi'(s) = \arg\max_{a} Q^{\pi}(s,a).$$

Temporal Difference (TD) learning is one of the most popular algorithms for estimating the value function. TD Learning has the advantage of being able to learn while an episode is running. According to the definition of the value function, TD learning is a method of minimizing the TD error in Equation (2).

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t), \tag{2}$$

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t. \tag{3}$$

Since fab simulation takes a long time to run one episode, TD learning is an attractive method which enables to learn during the simulation. There are popular methods based on TD learning such as SARSA and Q-learning. In this paper, SARSA(λ) learning is used. For more information, see Reinforcement Learning : An Introduction (Sutton 2018)

3.2 Reinforcement Learning for Semi-Markov Decision Process

Classical MDP formulation assumes that the time horizon T is discrete. But, in the actual fab, schedules are made by a sequence of decisions on the continuous time horizon. Since the time interval between decision makings is not constant, RL algorithms introduced in Section 3.1 cannot be applied directly. This case can be seen as Semi-MDP and related studies were done by Bradtke and duff (1995). The study introduced a way to apply the RL to Semi-MDP. According to their research, the discount factor γ and reward terms

r_{t+1} used for the MDP are changed to form of depending on the time interval τ between decision making. These changes make semi-MDP solution reach to the optimal policy as classical MDP does.

$$r_{t+1} \rightarrow \int_0^{\tau} e^{-\beta k} r_{t+k} dk, \tag{4}$$

$$\gamma \rightarrow e^{-\beta\tau}. \tag{5}$$

TD error in Equation (2) is for MDP. Therefore, according to Equation (4), (5), Equation (2) can be re-written as follows:

$$\delta_t = \int_0^{\tau} e^{-\beta k} r_{t+k} dk + e^{-\beta\tau} V(s_{t+\tau}) - V(s_t)$$

3.3 Function approximation and State Feature Selection for Fab

The manufacturing system has a large state space and action space. The state space contains information about entire buffers and machines as shown in following equations:

$$s = [\vec{B}, \vec{M}]$$

$$\vec{B} = \left[\vec{B}_1, \vec{B}_2, \dots, \vec{B}_n\right] \quad \vec{B}_i = \left[\vec{J}_1, \vec{J}_2, \dots, \vec{J}_{c_i}\right] \quad \vec{J}_j = [Arr_j, Due_j, Proc_j, Wait_j, Tool_j, \dots]$$

$$\vec{M} = \left[\vec{M}_1, \vec{M}_2, \dots, \vec{M}_m\right] \quad \vec{M}_k = [WaitingJobs_k, Loading_k, LastJobType_k, LastToolType_k, \dots]$$

$\vec{B}$ is the state vector of all buffers and n is the total number of buffers. Each vector $\vec{B}_i$ consists of job property vectors $\vec{J}_j$ where c_i is a maximum capacity of buffer i. A vector $\vec{J}_j$ has job information such as arrival time, due date, process time, waiting time in a buffer, tool type and so on. $\vec{M}$ is the state vector of all machines and m is the total number of machines. $\vec{M}_k$ holds information about the jobs that are currently waiting in buffers and current or last loaded job of machine k.

To estimate the value function of such large systems, parameterized function approximation is considered. The value function can be represented as a general, linear, non-linear function or neural net structure with appropriate state features depending on parameter vector $\vec{\theta}$.

$$V(s) = \vec{\theta}^T \vec{\phi}(s) \; or \; g(\vec{\phi}(s), \vec{\theta}) \tag{6}$$

Using function approximation, the value function update in Equation (3) is changed to the update form in the parameter space as show in Equation (7). Since large scale approximate dynamic programming (i.e reinforcement learning) assumes that the state transition probability of target system is unknown, various Stochastic Gradient Methods (SGD) such as Adam and RMSProp are useful for sample backup.

$$\vec{\theta} \leftarrow \vec{\theta} + \alpha\delta\nabla_{\vec{\theta}} V(s) \tag{7}$$

While approximation of the value function eliminates the concern on large state space, it leaves the problem of setting the state features well. According to Taylor (2011), good state features are able to capture the reward structure of the target system and predict next state features well. Furthermore, since the state features are subsets of the entire state space, the state features should contain lots of information about the target system. For example, rather than expressing the entire fab as a number of workpieces, it is better to express an entire fab in terms of the number of workpieces and their waiting time, processing time, and the number of product types. And in the case of reward, when the objective is to minimize the setup count, it

is better to give the information whether the workpiece will cause the setup or not rather than how late the workpiece is. To satisfy these conditions, we have considered dispatching scores introduced in Section 2.2 and Li et al. (2012; 2013) as state features:

$$\vec{\phi}(s) = \begin{bmatrix} \vec{\phi}^1(s) \\ \vec{\phi}^2(s) \\ \vec{\phi}^3(s) \\ \vec{\phi}^4(s) \\ \vec{\phi}^5(s) \\ \vdots \\ \vec{\phi}^N(s) \end{bmatrix} \qquad \vec{\phi}^i(s) = \begin{bmatrix} F^{i,1}(s) \\ F^{i,2}(s) \\ F^{i,3}(s) \\ F^{i,4}(s) \\ F^{i,5}(s) \\ \vdots \\ F^{i,M_i}(s) \end{bmatrix} \tag{8}$$

Dispatching scores made by the fab operator for decision making can adequately represent the state of the fab and react exactly to the reward structure to achieve. In Equation (8), $\vec{\phi}^i(s)$ denotes a vector of dispatching scores $F^{i,j}(s)$ of the last selected job in the machine i, M_i is the number of dispatching scores related to the machine i and $\vec{\phi}(s)$ is a vector of $\vec{\phi}^i(s)$ for all machines. Since the state transition probabilities of the system are unknown, the next state s' is also unknown. But, with this state feature vector, the state feature vector right after decision making is definitely known because we know that the dispatching score vector $\vec{\phi}^i(s)$.

$$\vec{\phi}(s') = \vec{\phi}(s^p) = \vec{\phi}(S^A(s,a)) \tag{9}$$

In Equation (9), the s' is the next state and s^p is the state after decision making at the current state s. s' and s^p may not be equal, but their state feature vectors are equal. This technique is based on the post-decision state variable introduced in Approximate Dynamic Programming: Solving the curses of dimensionality (Powell 2007)

And also this approach retains the concept of weight sum method of the real fab operators. If assumed that the function for approximation is linear, the way to select is exactly the same as the operator's method. But the meaning of weights used for decision making changes into terms of expectation of discounted reward sum and the weights are automatically adjusted by RL algorithm under a suitable reward structure, not human. In the next section, we will discuss a suitable reward structure for fabs

3.4 Reward Structure for Fabs

Fab schedules are evaluated in various ways such as production amount, machine utilization, due date violation, setup count, etc. In order to set rewards that satisfy various purposes at the same time, it is preferable to have no strong correlation between KPIs that are to be achieved. If the reward set for improving the KPI is greatly overlapped, the variance in the learning steps may be adversely affected, resulting in a wrong direction or slow learning. For example, it is not good to set up rewards related to the increase in production and in the utilization of machines at the same time. That is because the higher the utilization of machines, the higher the production volume, and the higher utilization of machines, the higher the production volume.

And also it is better to use intuitive reward structure, rather than abstracted one. In the previous researches, WIP level and Cycle time of the jobs were KPIs. Minimizing the WIP level improves the overall performance of the fab, but it does not lead to a specific performance improvement designed by the fab operator. Achieving improvement of desired performances is important because the fab policies can vary depending on demand and market conditions. By tuning the reward structure, a proper schedule for rapidly changing market conditions can be obtained through the proposed RL method.

3.5 TD Learning Algorithm for Semi MDP Formulated Fab Dispatching Problem

First, it needs to initialize parameters to be used for learning and start the fab simulation. Then, add the appropriate reward to the list $\mathcal{R}$ of the reward at the event time to generate the reward. At the point of dispatching event during simulation, obtain τ from the time of previous dispatch event and calculate the reward and discount rate based on τ. Calculated the scored dispatch priorities for all selectable jobs in the $A(s)$ and get $\mathcal{V}$ which is the set of all value function of post-decision state. Select the job from $\mathcal{V}$ through the suitable selection method to be used for learning. After selection, obtain V and V' from the $\vec{\phi}$ and $\vec{\phi}'$ calculated in the previous dispatching event, and use them with r and γ to find the TD error δ. If λ is not zero, eligibility trace vector $\vec{e}$ also need to be updated. Since the function approximation TD learning based on the sample gradient, update the $\vec{\theta}$ using the appropriate SGD method. Finally, replace the state feature vector $\vec{\phi}$ to $\vec{\phi}'$ and clear the reward list $\mathcal{R}$. Repeat the learning procedure until the end of the simulation. In the next section, the experimental results obtained from the virtual fab is shown.

Algorithm 1: SARSA(λ) Algorithm for fab scheduling.

Initialize : $\vec{\theta}$ arbitrarily, $\vec{\phi}$ appropriately, and $\vec{e}=\vec{0}$

Repeat(Until simulation ends)
  Repeat(for each reward event)
    $\mathcal{R} \leftarrow$ set of rewards
  Repeat(for each dispatching event)
    $s \leftarrow$ current state of simulation
    $\tau \leftarrow$ the time interval between previous and current dispatching event
    $\gamma \leftarrow e^{-\beta\tau}$
    $r \leftarrow$ sum of all discounted rewards in $\mathcal{R}$

    For all i $\in A(s)$
      $\mathcal{V} \leftarrow V(S^A(s,i))$
    $a \leftarrow$Select i from $\mathcal{V}$ by using suitable selection method
    $\vec{\phi}' \leftarrow \vec{\phi}(S^A(s,a))$
    $V \leftarrow g(\vec{\theta},\vec{\phi})$
    $V' \leftarrow g(\vec{\theta},\vec{\phi}')$
    $\vec{e} \leftarrow \gamma\lambda\vec{e} + \nabla_{\vec{\theta}}V$
    $\delta \leftarrow r + \gamma V' - V$
    $\vec{\theta} \leftarrow$ Update with δ and $\vec{e}$ by using suitable SGD method
    $\vec{\phi} \leftarrow \vec{\phi}'$
    Clear $\mathcal{R}$
End

4 EXPERIMENTS

4.1 Experimental Environment

The virtual fab to be used in the experiment is a fab similar to the LCD fab that is currently running. The fab has a total number of 160 machines and consists of numerous Clean, Photo lithography, CVD units and others. Each of machines has averagely 5 types of dispatching scores for decision making and the total number of dispatching scores in the entire fab is 682. The fab produces 50 kinds of products, and the demand for the products is consisted of 6 types and 250 jobs per day on average. It is assumed that the product arrives at the fab is not random and follows a predefined demand scenarios. The number of processes that

needed in the production varies from product to product, averagely 140 processes per product are required to be visited until leaving the fab as a finished product. We used the MozArt (Ko et al. 2013) which is a fab simulation engine, developed by VMS Solutions Co., Ltd to simulate the virtual fab.

The selection method used for learning is ε-greedy. With the probability (1-ε), the highest scored task is selected, otherwise, it is selected randomly. The SGD method used to update the parameter is Adam (Kingma and Ba 2014) which is widely used in academia recently. This method minimizes the adverse effects of learning from different scales of features and is a momentum-based method which enables quick searches for local solutions.

We focused on three objectives: increase production volume, preventing due date violation, and reducing the setup count of photo machines which becomes bottle neck process steps, to improve virtual fab. The reason behind setting such objectives is because they are intuitive and most widely used KPIs in the real LCD fabs. And also the intuitive reward is better than the abstracted reward for improvement of specific KPIs. To achieve the objectives, a reward for the finished product is set to 1000 per product, -100 per late product per day and -500 per photo machine setup change. The numerical values are arbitrarily set by the experimenter and will be discussed in detail in Section 5.

4.2 Experiment 1

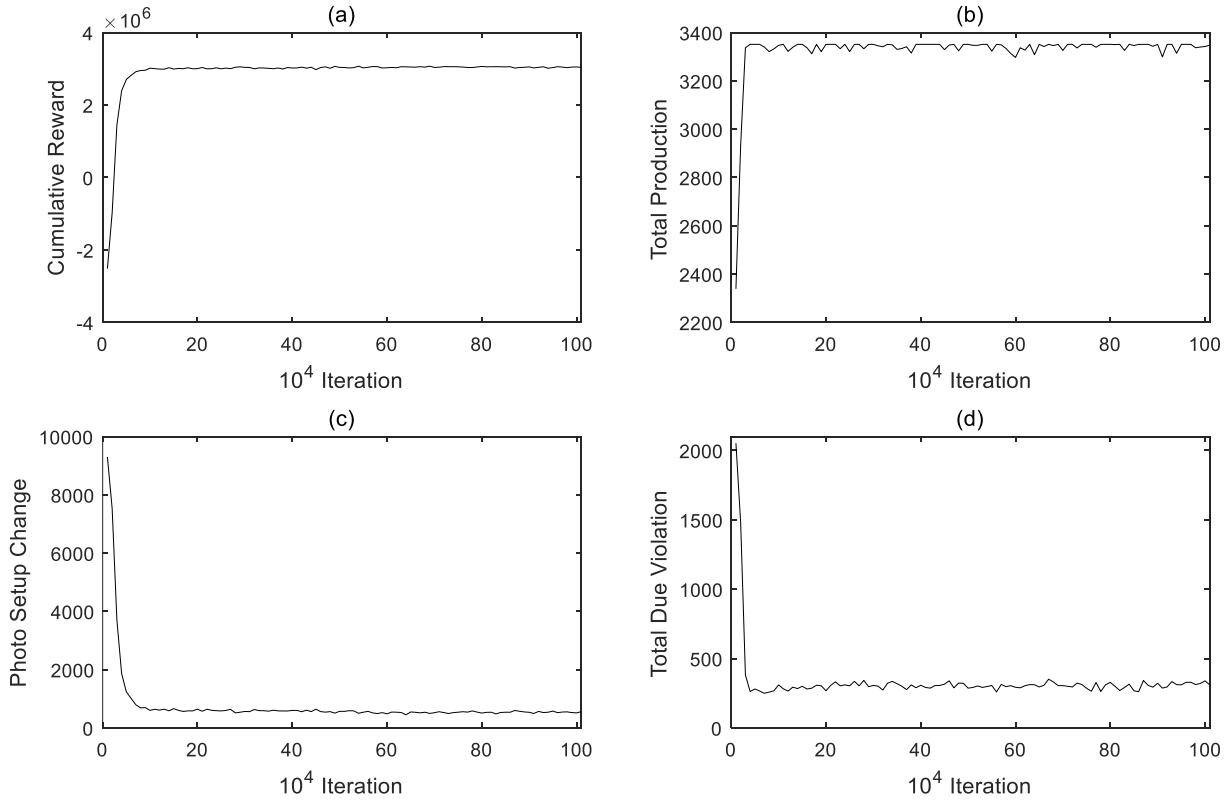

Figure 3 : KPIs at every 10000 iteration.

The goal of this experiment is to see whether a schedule obtained from RL is performing better than a schedule at the start of learning. A schedule with high cumulative reward value is better than a schedule with low cumulative reward value under the same reward structure. So, if RL works, cumulative reward will increase as learning progresses. For the experiment, the initial parameter vector $\vec{\theta}$ was all given equal to -1, and the approximation of the value function is simply linear in Equation (6). Of course, the function

structure for approximation can be replaced to Neural net or other general functions, and it can be considered as a future research work. The following is the result of learning the virtual fab with a fixed 14-days demand scenario (i,e. a job arrives at a fixed time as the real fab does)

In the simulation with the given scenario, about 68000 learning iteration steps occur. When the scenario ends, run the same scenario again using $\vec{\theta}$ obtained from the previous scenario. The graphs show the experiment results of simulation for 14 days with $\vec{\theta}$ obtained from every 10^4 learning iteration steps. This is because of the use of ε-greedy for learning and the use of greedy for evaluation. From the experiment result (a), it can be clearly seen that the cumulative reward value increases as the iteration proceeds. The graphs (b),(c) and (d) show the improvement of each KPI. As learning progresses, the total amount of production increase, the total number of setup changes in photo machines decrease and the total amount of due date violations decrease.

Table 2: Experiment summary of KPIs.

	Human	Initial	Avg RL	Best RL
Cumulative Reward	2891970	-2515706	3044316	3072052
Total Production	3292	2339	3342	3351
Photo Setup Change	740	9299	535	497
Due Date Violation	324.42	2052.06	304.29	304.47

Table 1 shows the overall KPIs from the experiment. The Column named “Human” is the result of the best schedule created by the VMS Solutions Co., Ltd engineer, "Initial" is the result at the start of the learning, "Avg RL" is the average of the results after half-million iterations of the learning, and "Best RL" is the best result during entire iterations. The average cumulative reward of RL is significantly higher than the cumulative reward of the initial schedule and also higher than the human-tuned schedule. It means that the schedule created by RL is on average better than the schedule created by human. In addition to cumulative reward, other indicators also improved.

4.3 Experiment 2

In Section 4.2, we showed that RL can improve desired KPIs under a purposeful reward structure. But, the portion of rewards is arbitrarily adjected by users. Therefore, it is necessary to check how the schedules are affected by the change of rewards structure. In this experiment, except for the due date violation penalty, the other settings were the same as the previous experiment. To see how the learning direction has changed, we set the due date violation penalty to -1000 which was -100 in the previous experiment.

Table 3: Average performance of experiment 1 and 2.

	Experiment 1	Experiment 2
Total Production	3344.34	3342.33
Photo Setup Change	535.19	551.31
Due Date Violation	304.29	287.79

Each column shows the average performance of the two experiments. Since the amount of the penalty increased, the amount of due date violation decreased. But, the numbers of photo setup change are increased. It seems that the photo units intentionally cause setup changes to avoid huge due date violation penalty. As we intended, this result exactly shows that the changes in the reward structure affect performance. The direction of learning can be controlled by adjusting the ratio of each reward with trade-offs. Still, there is a part that human has to participate, it is not necessarily disadvantaged.

4.4 Experiment 3

In the previous sections, the experiments were conducted under a given fixed-demand scenario. In this section, we check how the policies from RL are robust in changes of fab situation and demand scenario. Total of four different scenarios are prepared for this experiment. Some of them are similar to the previous scenario and some of are totally different from the previous one. The demand set 1(D_1) is the demand set used in Experiment 1 and 2. D_2 is a demand set which is obtained from 1-day simulation of D_1. In the same way, D_3 is obtained from 2-day simulation of D_1. D_4 is randomly generated scenario with 80% arrival level of D_1. The following table shows the cumulative rewards of the four demand scenarios under the policy obtained through reinforcement learning in the D_1 & D_2 and the policy created by the engineer.

Table 4: Comparison of the policies from different learning scenario.

	Human	Best RL D_1	Best RL D_2
Experiment D_1	2891970.94	3072052.30 (+6.22%)	3033323.961 (+4.89%)
Experiment D_2	2809619.48	2927850.64 (+4.20%)	2956354.36 (+5.22%)
Experiment D_3	2649341.72	2749867.80 (+3.79%)	2750446.54 (+3.82%)
Experiment D_4	2456494.88	2501844.53 (+1.84%)	2441535.67 (-0.06%)

The parentheses in Table 4 show the increase/decrease amount of cumulative reward compare to human policy. Except one result comes from the "Experiment D_4" with best RL policy trained through D_2, the others are all positive. It means that policies obtained through RL are better than human policy in most of the given scenario. Another interesting observation is that each of RL policy has relatively high performance gap in its learning scenario, which means that RL ensures a good schedule in a learning scenario, but not in others. It is critical issue that reinforcement learning gives better schedules than human adjusted schedules, but only in scenarios similar to those used at the time of learning. In other words, the proposed method is not robust to various scenarios. This is one of the limitations in this research. Details will be discussed in Section 5.

5 LIMITATIONS

We suggested a RL approach to learning the fab scheduling. However, there are still some limitations. First, the features that we suggested have been designed to process a given demand well, however, when there are changes in the demand, a good schedule is not always guaranteed. As in the experiment results in Section 4.4, when there are similarities between the demands, the results also show similar tendencies in improvement, But when the demands are changed, there is possibility of bad schedules. This is because the ranges of state features vary on scenarios and there are no features that can capture changes in demand. If an arbitrary state feature element $F^{i,j}(s)$ in scenario D_1 experiences only in range (0.4,0.6) and the same

feature in scenario D_1 experiences only in range (0.1,0.3), then the directions of learning are different. And also the parameters from the two scenarios function differently.

The second limit is, as mentioned in Section 4.3, that there are still some parts where human engagement is needed. Even if the parameters for decision making is automatically tuned through RL algorithm, the reward structure should be set by the human to improve desired objectives. The effort to set proper rewards may be painful.

6 CONCLUSION

In this paper, a fab scheduling problem is formulated into a semi-MDP and the RL approach with the state feature construction based on weight sum concept is proposed. The results for the virtual LCD fab show that the multiple performance indicators can be improved through RL at the same time. As explained in the limitations, rewards still need to be adjusted. However, the value of this research is that it has reduced the number of weights that need to be adjusted. Although the rewards still need to be set manually, through this research, the problem of setting hundreds of dispatching scores is transformed into a problem which requires setting a small number of rewards. The manual setting of the rewards is not necessarily a disadvantage since the priorities can be changed anytime by replacement of the operators and change of the fab policies.

The use of the proposed RL algorithm is limited but useful for the real fab operations. Since the method provides a good schedule for a given or similar demand set, if learning is conducted every time demand changes, good schedules can be provided repeatedly. And also the method of constructing state features does not depend on the system size so that, we can apply the method to the larger systems like semi-conductor fabs if the computation power is sufficient.

REFERENCES

Bradtke, S. J. and M. O. Duff. 1995. "Reinforcement learning methods for continuous-time Markov decision problems". *Advances in neural information processing systems,* November 27th-December 2nd, Denver, Colorado, USA, 393-400

Chen, X., X. Hao, H. W. Lin, and T. Murata. 2010. "Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning". *2010 IEEE International Conference on Automation and Logistics,* August 16th -August 20th, Hong Kong and Macau, China, 396-401

De Farias, D. P. and B. Van Roy. 2000. "On the existence of fixed points for approximate value iteration and temporal-difference learning". *Journal of Optimization theory and Applications* 105(3):589-608.

Đurasević, M. and D. Jakobović. 2018. "A survey of dispatching rules for the dynamic unrelated machines environment". *Expert Systems with Applications* 113:555-569.

Gosavi, A. 2004. "Reinforcement learning for long-run average cost". *European Journal of Operational Research* 155(3):654-674.

Haupt, R. 1989. "A survey of priority rule-based scheduling". *Operations-Research-Spektrum* 11(1):3-16

Kingma, D. P. and J. Ba. 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv*:1412.6980.

Ko, K., B. H. Kim, and S. K. Yoo. 2013. "Simulation based planning & scheduling system: MozArt®". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 4103-4104. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kumar, P. R. 1993. "Re-entrant lines". *Queueing systems* 13(1-3):87-110.

Lee, J., Y. Kim, J. Kim, Y. B. Kim, H. J. Kim, B. H. Kim and G. H. Chung. 2018. "A framework for performance analysis of dispatching rules in manufacturing systems". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 3550-3560. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Li, L., Z. Sun, M. Zhou, and F. Qiao. 2012. "Adaptive dispatching rule for semiconductor wafer fabrication facility". *IEEE Transactions on Automation Science and Engineering* 10(2):354-364.

Li, L., S. Zijin., N. Jiacheng, and Q. Fei. 2013. "Data-based scheduling framework and adaptive dispatching rule of complex manufacturing systems". *The International Journal of Advanced Manufacturing Technology* 66(9-12):1891-1905.

Park, B. C., E. S. Park, B. K. Choi, B. H. Kim and J. H. Lee. 2008. "Simulation based planning and scheduling system for TFT-LCD fab". In *Proceedings of the 2008 Winter Simulation Conference,* edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 2271-2276. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Powell, W. B. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Hoboken, New Jersey: John Wiley & Sons

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. Hoboken, New Jersey: John Wiley & Sons

Ramirez-Hernandez, J. A. and E. Fernandez. 2007. "Control of a re-entrant line manufacturing model with a reinforcement learning approach". In *Sixth International Conference on Machine Learning and Applications,* edited by M. A. Wani, M. M. Kantardzic, T. Li, Y. Liu, L. Kurgan, J. Ye, M. Ogihara, S. Sagiroglu, X. W. Chen, L. Peterson and K. Hafeez, 330-335. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ramírez-Hernández, J. A. and E. Fernandez. 2009. "A simulation-based approximate dynamic programming approach for the control of the intel mini-fab benchmark model". In *Proceedings of the 2009 Winter Simulation Conference,* edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1634-1645. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Shahrabi, J., M. A. Adibi, and M. Mahootchi. 2017. "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling". *Computers & Industrial Engineering* 110:75-82.

Sutton, R. S. and A. G. Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Taylor, G. 2011. *Feature Selection for Value Function Approximation*. Durham, North Carolina, USA, Duke University.

AUTHOR BIOGRAPHIES

WONJUN LEE is a Senior Engineer of VMS Solutions Co., Ltd.. He received BS and MS from Korea Advanced Institute of Science and Technology(KAIST) all in industrial engineering. His recent research interests are modeling & simulation, fab scheduling and machine learning, especially reinforcement learning. His email address is wjlee@vms-solutions.com.

BYUNG-HEE KIM is the President of VMS Solutions Co., Ltd.. He received a BS from Sungkyunkwan University, MS and Ph.D. from Korea Advanced Institute of Science and Technology(KAIST) all in industrial engineering. He is interested in simulation-based scheduling and planning, manufacturing information systems, BPMS, and virtual manufacturing. His email address is kbhee@vms-solutions.com.

KEYHOON KO is the President of VMS Global, Inc. located in Vienna, VA, USA. He holds a bachelor's degree in industrial engineering from Sungkyunkwan University, Korea. He also earned a master's degree and Ph.D. in industrial engineering from Korea Advanced Science and Technology (KAIST), Korea. He joined VMS Solutions Co Ltd as a founding member in 2000. He had led technical research center for 14 years and is currently responsible for all areas of global business including marketing and sales in America and Europe. His experiences on virtual manufacturing cover die and mold, automotive, shipbuilding, display panel and semi-conductor industries. His email address is fish@vms-solutions.com.

HAYONG SHIN is a professor in the Department of Industrial & Systems Engineering at KAIST (Korea Advanced Institute of Science and Technology). Before joining KAIST, He worked for Chrysler Corp., CubicTek Co. and LG Electronics, developing commercial and in-house engineering software solutions. He received a BS from Seoul National University in 1985, an MS and a PhD from KAIST in 1987 and 1991, all in industrial engineering. His recent research interests are in the area of machine learning, simulation, geometric modeling for manufacturing system applications. His email address is hyshin@kaist.ac.kr