Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

GRAPH-BASED REINFORCEMENT LEARNING FOR DYNAMIC PHOTOLITHOGRAPHY SCHEDULING

Sang-Hyun Cho[1], Sohyun Jeong[2], Jimin Park[1], Boyoon Choi[3], Paul Han[3], and Hyun-Jung Kim[1]

[1]Dept. of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon, REPUBLIC OF KOREA
[2]Graduate School of Data Science, Korea Advanced Institute of Science and Technology, Daejeon, REPUBLIC OF KOREA
[3]Mechatronics Technology Research Center, Samsung Display Company Ltd., Yongin, Gyeonggi, REPUBLIC OF KOREA

ABSTRACT

This paper addresses the photolithography process scheduling problem, a critical bottleneck in both display and semiconductor production. In display manufacturing, as the number of deposited layers increases and reentrant operations become more frequent, the complexity of scheduling processes has significantly increased. Additionally, growing market demand for diverse product types underscores the critical need for efficient scheduling to enhance operational efficiency and meet due dates. To address these challenges, we propose a novel graph-based reinforcement learning framework that dynamically schedules photolithography operations in real time, explicitly considering mask locations, machine statuses, and associated transfer times. Through numerical experiments, we demonstrate that our method achieves consistent and robust performance across various scenarios, making it a practical solution for real-world manufacturing systems.

1 INTRODUCTION

In recent years, OLED display demand has surged dramatically across mobile phones, televisions, automotive displays, and various other applications (Huang et al. (2020)). This market expansion, coupled with continuous technological advancements, has significantly increased manufacturing complexity. In particular, the fabrication of thin-film transistor (TFT) backplanes—components that control individual pixels in OLED panels—has become particularly sophisticated (Ji et al. (2021)). Modern TFT backplanes require numerous deposited layers, including various thin films, electrodes, and insulating materials, to achieve higher resolutions and advanced pixel structures (Sun et al. (2023)). As layer complexity increases, manufacturing processes increasingly involve reentrant flows, where specific operations must return to the same equipment multiple times, further complicating production scheduling and management.

While TFT fabrication encompasses multiple processes, including deposition, cleaning, photolithography, and etching, we concentrate specifically on photolithography as it represents the most significant production bottleneck (Ghasemi et al. (2020)). This critical process uses light to transfer intricate circuit patterns onto substrates using highly specialized and expensive equipment. Consequently, optimizing this stage directly impacts overall production efficiency. However, scheduling photolithography operations presents unique challenges beyond standard parallel-machine scheduling problems, as it requires simultaneous management of both machines and photomasks. These photomasks must be precisely tracked in real time to ensure compatibility with specific jobs, with production timing dependent on both machine availability and mask readiness.

Our research addresses these challenges by developing methods to rapidly generate high-quality schedules for photolithography operations in dynamic production environments. In practice, photolithography tasks arrive continuously, creating dual-resource constraints where both appropriate machines and required masks

must be simultaneously available. This introduces additional complexity through operational factors such as mask transport times and setup requirements. By integrating these elements into our scheduling approach, we aim not only to enhance overall OLED manufacturing throughput but also address practical production objectives, including prioritizing high-value products and meeting specific step targets by designated deadlines. Furthermore, our real-time tracking system for mask locations and machine statuses enables dynamic scheduling of incoming tasks, resulting in more efficient, responsive, and adaptable production processes.

There are many studies that have explored photolithography scheduling using a variety of modeling and optimization techniques. For example, Cakici *et al.* (2007) introduced a network-based optimization model for parallel machine scheduling under auxiliary resource constraints, proposing heuristics that yield near-optimal solutions for small-scale instances and refining them via Tabu search. Hung *et al.* (2013) investigated rescheduling in semiconductor wafer fabrication, developing a "sensitivity search" strategy to handle disturbances such as machine breakdowns and mask availability. Zhang *et al.* (2018) reduced total completion time by combining a rolling horizon approach with an improved imperialist competitive algorithm, while Deenen *et al.* (2023) modeled photolithography scheduling as an unrelated parallel-machine problem and outperformed mixed-integer programming methods. More recently, Kim *et al.* (2023) introduced a reinforcement learning (RL) approach for AMOLED photolithography. However, their deep Q-network still requires retraining whenever mask counts or product types change, limiting its adaptability in highly dynamic settings.

Graph-based methodologies have become increasingly common in the broader field of neural combinatorial optimization (NCO), where deep learning techniques are applied to tackle large-scale, evolving combinatorial problems (Park, Bakhtiyar, and Park 2021). In diverse scheduling tasks—from crane operations and parallel-machine environments to directed acyclic graph-based workflows—RL has shown particular promise. For instance, Liu *et al.* (2023) presented a deep Q-network-based meta-dispatching rule that adapts to both current and historical system states, while Chang *et al.* (2022) developed a double deep Q-network with a soft ε-greedy policy for random job arrivals in flexible job shops. Similarly, Cho *et al.* (2024) introduced a real-time RL-based scheduling algorithm for dynamic steel coil warehouses where multiple cranes share a single track, and Ni *et al.* (2021) proposed a Multi-Graph Attributed Reinforcement Learning–based Optimization (MGRO) algorithm for large-scale hybrid flow shop scheduling, effectively leveraging multi-graph representations and reward shaping to adapt to real-time changes.

Despite these advances, few studies have addressed the unique combination of mask sharing and machine-specific constraints inherent to photolithography. To bridge this gap, we propose a size-agnostic, graph-based RL framework that accommodates mask sharing and reentrant flows, efficiently tackling the dynamic nature of photolithography lines without incurring substantial retraining overhead when production settings change.

2 PROBLEM FORMULATION

2.1 Scheduling Problem and Overall Approach

We consider a scheduling problem involving n jobs, m photolithography machines, and l masks, where each job belongs to one of L distinct lot types. At time zero, an initial set of n_{init} jobs is already present in the system, while the remaining jobs arrive in real time according to a specified inter-arrival distribution. Each job j is assigned a weight w_j that indicates its priority, reflecting factors such as urgency or overall importance. Additionally, each job j can be processed only on a designated subset of machines $\mathscr{M}_j \subseteq \{1,2,\ldots,m\}$ (i.e., machine eligibility constraints). For every eligible job-machine pair (j,k) where $k \in \mathscr{M}_j$, job j is assigned a specific mask $\mu_{jk} \in \{1,2,\ldots,l\}$ with a processing time p_{jk}. By explicitly incorporating mask allocation, mask transfer, and machine compatibility into our formulation, we aim to generate efficient schedules for the photolithography process that effectively address these interdependent resource constraints. Figure 1 provides a simple Gantt chart example that illustrates the complexity of

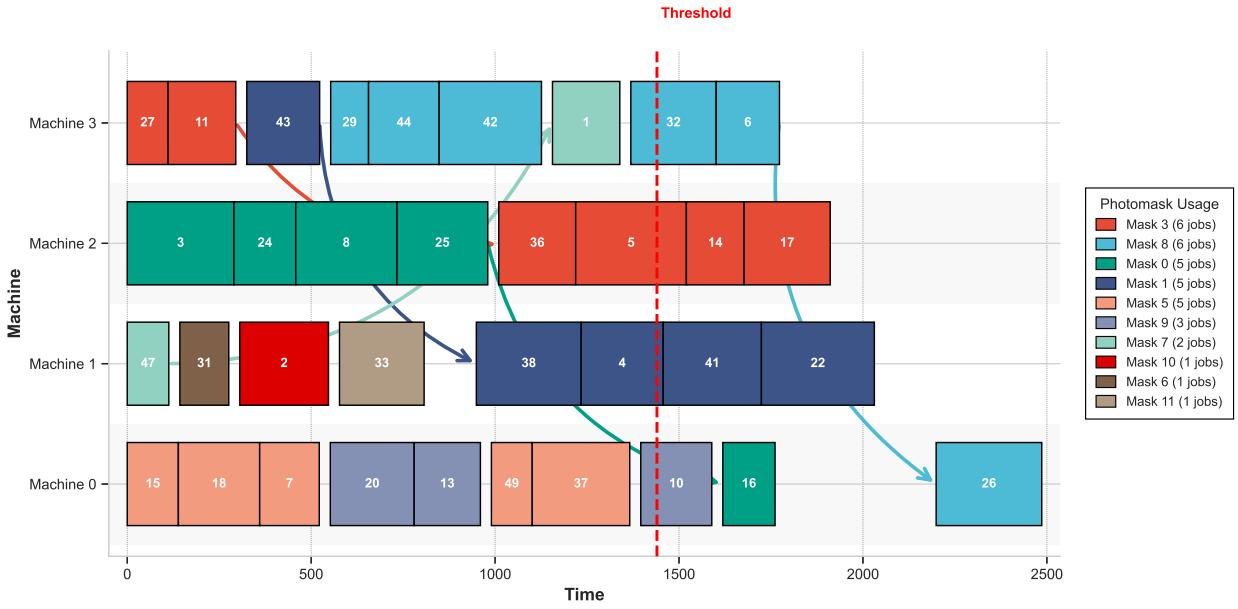


Figure 1: Gantt chart illustrating job scheduling with mask transfers

scheduling when both mask allocation and machine assignment must be considered simultaneously, showing a scenario with four photolithography machines where jobs require specific masks that may need to be transferred between machines.

2.2 Key Performance Indicators (KPIs) and Aggregated Objective

To evaluate the quality of a schedule, we employ three Key Performance Indicators (KPIs) over a fixed time horizon h. Let $T = h$ denote the end of this horizon, and let C_j be the completion time of job j. Our KPI formulation builds upon the approach presented by Kim *et al.* (2023), which originally introduced three metrics: throughput, step-target fulfillment, and setup count. Since setup reduction is ultimately reflected in improved throughput, we omit the setup-related KPI and instead introduce a priority-oriented metric. Specifically, we define:

- **KPI_1 (Throughput)**:

$$\mathrm{KPI}_1 = \sum_{j=1}^{n} \mathbf{1}\{C_j \le T\},$$

  where $\mathbf{1}\{\cdot\}$ is an indicator function that counts a job if it finishes by T. Higher values of KPI_1 indicate better throughput within the horizon.
- **KPI_2 (Step Target)**:

$$\mathrm{KPI}_2 = \sum_{\ell=1}^{L} \mathrm{Deficit}_\ell,$$

  where $\mathrm{Deficit}_\ell$ denotes the shortfall in meeting the target number of lots for type ℓ by time T. Lower values of KPI_2 imply better fulfillment of each lot type's production target.
- **KPI_3 (Weighted Priority)**:

$$\mathrm{KPI}_3 = \sum_{j=1}^{n} \Big(w_j \max\{T - C_j, 0\} \Big),$$

where w_j is the priority weight of job j. The term $\max\{T - C_j, 0\}$ measures how much earlier job j completes relative to T; multiplying by w_j places greater emphasis on finishing higher-priority jobs earlier.

While each KPI can be evaluated independently, our primary goal is to optimize a weighted combination of these metrics:

$$\max\Big(\alpha_1 \cdot \mathrm{KPI}_1 \; - \; \alpha_2 \cdot \mathrm{KPI}_2 \; + \; \alpha_3 \cdot \mathrm{KPI}_3\Big),$$

where $\alpha_1, \alpha_2, \alpha_3 \geq 0$ denote user-specified weights reflecting the relative importance of throughput, step-target fulfillment, and weighted priority, respectively. Note that KPI_1 and KPI_3 are designed to be *maximized*, while KPI_2 is designed to be *minimized*; hence we assign a minus sign before $\alpha_2 \cdot \mathrm{KPI}_2$ in the objective. By jointly considering throughput (KPI_1), step-target fulfillment (KPI_2), and weighted priority (KPI_3), we achieve a balanced assessment of each schedule's effectiveness within the specified horizon $[0, T]$.

3 PROPOSED APPROACH

3.1 Overall Architecture

Our approach models the scheduling environment as a graph where nodes contain information on jobs, machines, and masks, while edges represent various relational constraints. A graph neural network (GNN) processes this state representation, extracting meaningful latent features from the raw node attributes. These features are then fed into a policy module that assigns probabilities to different job–machine–mask pair actions. The complete framework is trained end-to-end using policy gradient methods, thereby encouraging the selection of actions that lead to improved scheduling performance.

3.2 State

We represent the state as a graph $G_t(V_t, E_t)$ that captures the scheduling environment at decision step t. Figure 2 illustrates an example state, where job nodes (orange circles) and machine–mask nodes (blue circles) are connected by different types of edges to represent various relationships among resources. Each decision step corresponds to a point in the constructive scheduling process at which an assignment (i.e., a job to a machine-mask combination) is made, and the graph is updated accordingly. This graph-based state representation provides the foundation for our approach's size-agnostic property. By encoding the scheduling environment as a graph with relative node and edge features, our method can naturally handle varying numbers of machines, masks, and job types without requiring architectural modifications. The graph neural network processes these variable-sized graphs through local message-passing operations, enabling the learned policy to generalize across different problem scales.

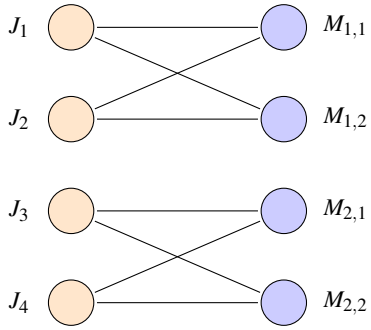
The node set V_t is partitioned into two subsets:

- **Job Nodes (V_t^a)**: Among all jobs that have arrived but are not yet completed (or scheduled) at time t, each job j corresponds to one *job node* v_j^a. This job node is associated with a feature vector

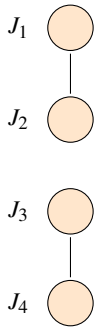
  $$\big[\mathbf{1}^{(\mathrm{job})},\, w_j,\, \ell_j,\, \tau_j,\, \hat{h}\big],$$

  where $\mathbf{1}^{(\mathrm{job})}$ is an indicator that this node is a *job node*, w_j is the priority (or weight) of job j, ℓ_j is the number of lots in job j, τ_j represents the remaining step targets for the lot type of job j, and $\hat{h} = \max(h - \mathrm{sim}_t, 0)$ denotes the time remaining until a certain horizon h, measured from the current simulation time.
- **Machine-Mask Nodes (V_t^b)**: For each machine-mask combination (k, ℓ) that can potentially process at least one of the *arrived and unscheduled job nodes*, we define a *machine-mask node* $v_{k\ell}^b$. This node is characterized by a feature vector

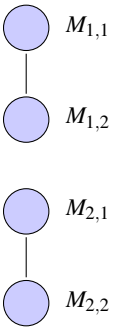
  $$\big[\mathbf{1}^{(\mathrm{pair})},\, m_{k\ell},\, \delta_{k\ell},\, r_{k\ell}^{\mathrm{mask}},\, r_k^{\mathrm{mch}},\, \hat{h}\big],$$



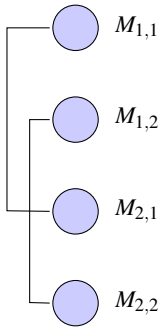
(a) Edges between job and machine–mask nodes (E^a).



(b) Edges among job nodes with the same lot type (E^b).



(c) Edges among machine–mask nodes for the same machine (E^c).



(d) Edges among machine–mask nodes for the same mask (E^d).

Figure 2: An example of graphical representations of states, illustrating a state comprising 4 jobs, 2 lot types, 2 machines, and 2 masks.

where $\mathbf{1}^{(\text{pair})}$ indicates that this node is a *pair (machine-mask)* node, $m_{k\ell}$ indicates whether mask ℓ is currently mounted on machine k, $\delta_{k\ell}$ is a binary value that equals 1 if mask ℓ was last used on machine k, $r^{\text{mask}}_{k\ell}$ represents the remaining time until mask ℓ is available (or done with setup/cleaning) on machine k, r^{mch}_{k} is the time until machine k completes its current job, and $\hat{h} = \max(h - \text{sim}_t, 0)$ again denotes the time remaining until the horizon h.

The edge set E_t is segmented into four subsets:

- **Edges between Job and Machine-Mask Nodes (E^a_t)**: An edge exists between a job node v^a_j and a machine-mask node $v^b_{k\ell}$ if job j can be processed on machine k with mask ℓ. The associated edge feature—such as the expected processing time—is denoted by p_{jk} when job j is compatible with both machine k and mask ℓ.
- **Edges among Job Nodes with the Same Lot Type (E^b_t)**: These edges connect job nodes v^a_j and $v^a_{j'}$ that belong to the same lot type. Since each lot type has a different required quantity by a specific time, and jobs in the same lot type share the same mask and machine, these edges highlight their common resource needs.
- **Edges among Machine–Mask Nodes for the Same Machine (E^c_t)**: These edges connect machine–mask nodes $v^b_{k\ell}$ and $v^b_{k\ell'}$ that belong to the same machine k. They capture transitions from

one mask ℓ to another ℓ'. Without these edges, we would lose the representation that these nodes share the same machine (Cai et al. (2021)).

- **Edges among Machine–Mask Nodes for the Same Mask (E_t^d)**: These edges connect machine–mask nodes $v_{k\ell}^b$ and $v_{k'\ell}^b$ corresponding to the same mask ℓ. Without these edges, we would lose the representation that these nodes share the same mask.

By updating $G_t(V_t, E_t)$ at each decision step t, we maintain a dynamic representation of the scheduling state. This graph-based formulation flexibly encodes complex constraints such as machine-mask assignments, job release times, and potential setup or transfer times, making it amenable to approaches based on graph neural networks or reinforcement learning.

3.3 Action, State Transition, and Reward

Each action involves selecting an available job node v_j^a and a machine–mask pair node $v_{k\ell}^b$ in the Photo stage. In other words, job j is assigned to the machine–mask pair consisting of machine k and mask ℓ. For job j, the start time S_j is defined as

$$S_j = \max\Big(r_j,\ \text{machineRelease}_k + \text{setup},\ \text{maskRelease}_\ell + \text{trans} + \text{setup}\Big),$$

where r_j is the arrival time of job j, machine release$_k$ is the time when machine k becomes available, mask release$_\ell$ is the time when mask ℓ becomes available, transportation time is the delay required to move mask ℓ to machine k (if needed), and setup is the setup time required before processing. Note that the mask's location and state are critical; if mask ℓ is already on machine k, no additional transportation time is needed, and if it was the last mask used on machine k, the setup time becomes 0.

If no feasible assignment exists at the current simulation time sim_t, this leads to a state transition in the scheduling system. Consequently, the event-driven simulation advances sim_t to the earliest time when at least one machine–job pair becomes available—that is, when a job is released or a machine becomes idle and meets the eligibility constraints. This ensures that the scheduling process proceeds continuously.

Once the scheduling process is complete—that is, when the simulation time exceeds a specified threshold—the reward is computed. At this terminal state, the reward is defined as the negative of an objective.

3.4 Graph Encoder

The graph encoder begins by transforming each node's raw feature vector x_i into an initial hidden representation:

$$h_i(0) = \text{ReLU}(W_0 \cdot x_i),$$

where W_0 is a trainable weight matrix and $h_i(0)$ is the initial embedding for node i.

The scheduling graph includes four distinct edge types, E_t^a, E_t^b, E_t^c, and E_t^d. We handle each corresponding subgraph separately using a GATv2-based attention mechanism (Brody *et al.* (2021)). For each edge type $k \in \{a,b,c,d\}$ and for each layer l (with $l = 0,\ldots,L-1$), the encoder computes an unnormalized attention score for an edge from node i to node j with associated feature f_{ij} (e.g., a setup time) as follows:

$$e_{ij}^k = a^k(l) \cdot \text{LeakyReLU}\Big(W_1^k(l) \cdot \big[h_i(l) \,\|\, h_j(l) \,\|\, f_{ij}\big]\Big),$$

where $a^k(l)$ is an attention vector, $W_1^k(l)$ is a learnable weight matrix, and $\|$ denotes vector concatenation. The attention coefficients are normalized via:

$$\alpha_{ij}^k = \frac{\exp(e_{ij}^k)}{\sum_{j' \in N_i^k} \exp(e_{ij'}^k)},$$

with N_i^k representing the set of neighbors of node i connected via edge type k.

Using these attention weights, the updated representation for node i with respect to edge type k is computed by:

$$h_i^k(l) = W_3^k(l) \cdot \text{LeakyReLU}\left(\sum_{j \in N_i^k} \alpha_{ij}^k \cdot \left(W_2^k(l)\, h_j(l) \right) \right),$$

where $W_2^k(l)$ and $W_3^k(l)$ are learnable parameters.

Finally, the representations from all four edge types are aggregated through concatenation and a residual connection to update the node embedding:

$$h_i(l+1) = \text{ReLU}\Big(h_i(l) + W_4 \cdot \left[h_i^a(l) \,\|\, h_i^b(l) \,\|\, h_i^c(l) \,\|\, h_i^d(l) \right] \Big),$$

where W_4 is a trainable weight matrix. This multi-layered attention mechanism is repeated for L layers, yielding final node representations $h_i(L)$ that capture both local interactions and the overall structure of the scheduling state.

3.5 Action Decoder and Probability Computation

The action decoder converts the final node embeddings into a probability distribution over possible scheduling decisions. For each candidate job–machine assignment, we concatenate the corresponding node embeddings and pass them through a feed-forward network:

$$y_i = W_6 \cdot \text{ReLU}\Big(W_5 \cdot \left[h_{\text{job}}^{(L)} \,\|\, h_{\text{pair}}^{(L)} \right] \Big) + \log(\text{mask}_i),$$

where the mask mask_i zeros out logits corresponding to infeasible actions (e.g., due to eligibility constraints). The logits y_i are normalized with a softmax function:

$$p_i = \text{Softmax}(y_i),$$

which yields a valid probability distribution over the feasible actions. During training, actions are sampled from this distribution, allowing the model to learn a scheduling policy that minimizes the overall objective.

3.6 Training Procedure

To train our proposed network, we adopt a policy learning approach based on the REINFORCE algorithm. Following the method in Mao *et al.* (2019), we define a baseline b as the mean cumulative reward at the final step of each episode. Let T_i denote the terminal time step of episode i; then the baseline is computed as

$$b = \frac{1}{M} \sum_{i=1}^{M} R_{T_i}^i.$$

This baseline is determined using an identical task arrival sequence, which reduces variance from random arrival times and enhances training stability in dynamic environments. The complete procedure is detailed in Algorithm 1.

4 COMPUTATIONAL EXPERIMENTS

4.1 Instance Generation

To evaluate our method, we generated synthetic instances that emulate real-world data using a custom data generation process. Processing times were sampled uniformly over the interval $[100, 300]$ and job weights were drawn from a uniform distribution over $[0.2, 1]$. Job arrivals followed a Poisson process

Algorithm 1 Policy Gradient Training Algorithm

Initialize policy parameters θ for π_θ.
for each update iteration **do**
  Set gradient accumulator $\Delta_\theta \leftarrow 0$ and sample a mini-batch of B instances.
  for each instance $j = 1, \ldots, B$ **do**
    Run M episodes with π_θ to collect trajectories $\{\tau^i\}_{i=1}^{M}$, where for each episode i, $\tau^i = \{s_1^i, a_1^i, \ldots, s_{T_i}^i, a_{T_i}^i, R_{T_i}^i\}$ with variable length T_i.
    Compute baseline $b \leftarrow \frac{1}{M}\sum_{i=1}^{M} R_{T_i}^i$.
    for each episode $i = 1, \ldots, M$ **do**
      Update gradient: $\Delta_\theta \leftarrow \Delta_\theta + \frac{1}{BM}\sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(s_t^i, a_t^i)\,(R_{T_i}^i - b)$.
    end for
  end for
  Update parameters: $\theta \leftarrow \theta + \alpha\,\Delta_\theta$.
end for

with rates proportional to the number of machines—approximately 25 jobs per 4-hour period in training, scaling linearly for test instances. In addition, the ratio of feasible machine–job pairs was set to 0.5, the transportation time—the time required for a mask to move between machines—was fixed at 360, and the setup time—the time required to change a mask on a machine—was set to 30. We adopt $\alpha_1 = 1$, $\alpha_2 = 0.1$, and $\alpha_3 = 0.25$ to ensure a balanced evaluation across the three KPIs while reflecting both their relative importance and scale differences in actual manufacturing settings. These weights prevent any single metric from dominating the objective, with throughput as the primary focus in line with manufacturing priorities.

4.2 Model and Training Parameters

In our experiments, the model is configured with an embedding dimension of 128, 8 attention heads, and 3 encoder layers. Each training instance is generated with 25 initial jobs, processed on 5 machines with 20 available masks, and the simulation terminates if the process exceeds a 4-hour time horizon. The training procedure is carried out over 100 iterations, each utilizing 1,500 instances (resulting in a total of 150,000 training instances) with a batch size of 50. For each problem instance, 6 schedules are generated; the average reward of these schedules is used as the baseline for reinforcement learning.

The training took approximately 12 hours, with smooth convergence and monotonically decreasing evaluation score. Despite training only on instances with 5 machines and 20 masks, our model successfully generalized to problem sizes ranging from 5-20 machines and 20-160 masks. All experiments were conducted on an Intel Core i9-14900KS CPU (64GB RAM) with NVIDIA RTX 4090 GPU (24GB).

4.3 Comparison Methods

4.3.1 Mask Setup Minimization Scheduling Heuristics

In our experimental setup, dispatching decisions are made on a per-machine basis at the moment a machine becomes idle. However, conventional dispatching rules, such as *SPT* (Shortest Processing Time) or *LPT* (Longest Processing Time), are not well-suited for our problem because they do not take into account the use of the mask or its current location, which is critical under photolithography constraints. Therefore, we propose a series of *Mask Setup Minimization Heuristics (MSMH)* that prioritize maintaining the same mask on a given machine, thereby reducing changeover times. We further integrate the MSMH principle with various tie-breaking criteria, including *SPT*, *Priority*, and *StepTarget*, to ensure that each rule aligns with a specific scheduling objective (e.g., minimizing flow time, prioritizing high-value jobs, or meeting step-target deadlines). All MSMH-based heuristics first look for a job that can use the mask last used on the machine. If such jobs exist, they apply the selection rule below; otherwise, they pick the job (from all

available) that achieves the earliest finish time (EFT) on the machine. The specific selection rule differs as follows:

- **MSMH–SPT**: Among the mask-compatible jobs, pick the one with the *shortest* processing time.
- **MSMH–Priority**: Among the mask-compatible jobs, pick the one with the *highest priority*.
- **MSMH–StepTarget**: Among the mask-compatible jobs, pick the one with the largest step-target deficit.

4.3.2 RL Approach

Following the fixed-dimension RL method proposed by Kim *et al.* (2023), we adopted their state representation as a baseline and then modified it to suit our KPI objectives. In our adaptation, we replaced the graph attention module with a standard multilayer perceptron (MLP) while retaining the same RL procedure—i.e., using REINFORCE with a learning rate of 0.0001 and the Adam optimizer. Specifically, we defined the state representation using four features: (1) the number of remaining lots per lot type, (2) the weight of each lot type (a feature not considered in the original method), (3) the remaining step-target amount per lot type, and (4) mask availability (this feature was slightly modified from the original to better align with our KPI). The MLP architecture consists of three layers, each with 256 hidden units and ReLU activation. Because the state vector is of a fixed size, this approach can only handle instances with predetermined dimensions.

To accommodate the baseline's limitation that no new mask or lot type be introduced, we trained this model on problem instances in which (i) the set of machines available per lot type and (ii) the mask associated with each lot type were fixed, while varying lot weights, release times, and step-target amounts. Under these constraints, the baseline can operate effectively. However, if the environment changes (e.g., a new mask is added), the trained baseline can no longer be applied. Hence, we generated problem instances adhering to these restrictions in order to compare the baseline with our proposed approach.

4.4 Experimental Results

We compared our graph-based RL algorithm with three MSMH heuristics, with results in Tables 1-4. We first tested a relatively low (baseline) arrival rate that aligns closely with the training distribution. Parameters such as the initial number of jobs n_{init}, the number of machines m, and the number of masks k were varied, but within ranges similar to the training set. As shown in Table 1, our RL-based method consistently outperformed the MSMH heuristics, demonstrating that the learned policy reliably produces high-quality schedules under conditions similar to those in which it was trained. Among the MSMH-based methods, MSMH–SPT showed a slight edge over both MSMH–Priority and MSMH–StepTarget, largely because selecting the shortest processing time tends to maximize the number of completed jobs, thereby indirectly improving other metrics (e.g., priority-related or step-target-related outcomes). Notably, our experiments include instances with up to 20 machines, which represents a practical scale comparable to many real-world photolithography workcenters in display manufacturing facilities. Figure 3 shows how the RL policy achieves high utilization while efficiently managing mask changes and minimizing setups (hatched areas) on a 20-machine instance.

Table 1: Performance Evaluation under Baseline Arrival Rate Conditions

$\mathbf{n}_{\text{init}}$	**m**	**k**	**MSMH–StepTarget**	**MSMH–SPT**	**MSMH–Priority**	**Ours**
25	5	20	7.99	9.18	8.00	17.82
50	10	40	23.01	25.22	23.77	35.40
100	20	80	73.92	78.15	74.58	83.81
Average			34.97	37.52	35.45	45.68

We then compared our approach with RL Fixed on 100 instances. Although RL Fixed was specifically tailored to fixed-dimension environments, our size-agnostic graph-based RL approach not only generalizes to instances of varying dimensions but also outperforms RL Fixed even under fixed conditions, as shown in Table 2. RL Fixed performs poorly due to insufficient state representation, particularly missing whether masks were last used on their current machines—critical information for minimizing setups.

Table 2: Performance Evaluation under Fixed Conditions: RL Fixed vs. Proposed RL Approach

n_{init}	m	k	MSMH–StepTarget	MSMH–SPT	MSMH–Priority	RL–Fixed	Ours
25	5	20	25.14	25.61	25.01	24.34	27.86

Subsequently, we evaluated performance under a significantly higher arrival rate—roughly twice as large as in the baseline case—to reflect a more demanding environment. In Table 3, our RL-based approach maintains its advantage under both light and heavy inflows, indicating robust generalization even when job arrivals deviate substantially from the training conditions.

Table 3: Performance Evaluation under High Arrival Rate Conditions

n_{init}	m	k	MSMH–StepTarget	MSMH–SPT	MSMH–Priority	Ours
25	5	20	5.43	5.78	5.31	12.90
50	10	40	19.94	22.75	19.70	36.50
100	20	80	90.54	95.68	90.61	116.10
Average			38.64	41.40	38.54	55.17

Finally, we examined scenarios with machine-to-mask and job-to-mask ratios that differ notably from those in the training distribution. Table 4 shows that our RL-based method continues to outperform the MSMH heuristics, reflecting the policy's ability to adapt to diverse capacity ratios without requiring additional retraining. These findings underscore the flexibility of our graph-based RL approach in accommodating different production conditions.

Table 4: Performance Evaluation under Diverse Job–Mask Ratios.

n_{init}	m	k	MSMH–StepTarget	MSMH–SPT	MSMH–Priority	Ours
50	10	20	6.51	8.84	3.43	13.35
50	10	80	36.79	38.51	38.15	43.57
100	20	40	13.75	13.96	8.16	19.88
100	20	160	89.09	91.76	91.34	89.93
Average			36.54	38.27	35.27	41.68

Overall, these results confirm that our framework performs robustly under diverse operating conditions, including varying arrival rates and different resource ratios.

5 CONCLUSION

In this paper, we presented a graph-based reinforcement learning framework for photolithography scheduling. Our approach models the scheduling environment with two distinct node types: job nodes and machine-mask pair nodes. This formulation accurately tracks real-time mask locations and job statuses within a unified graph structure. Furthermore, the scheduling policy is trained via reinforcement learning to adapt to dynamic job arrivals, and its size-agnostic formulation reduces the need for repeated retraining as system conditions change.

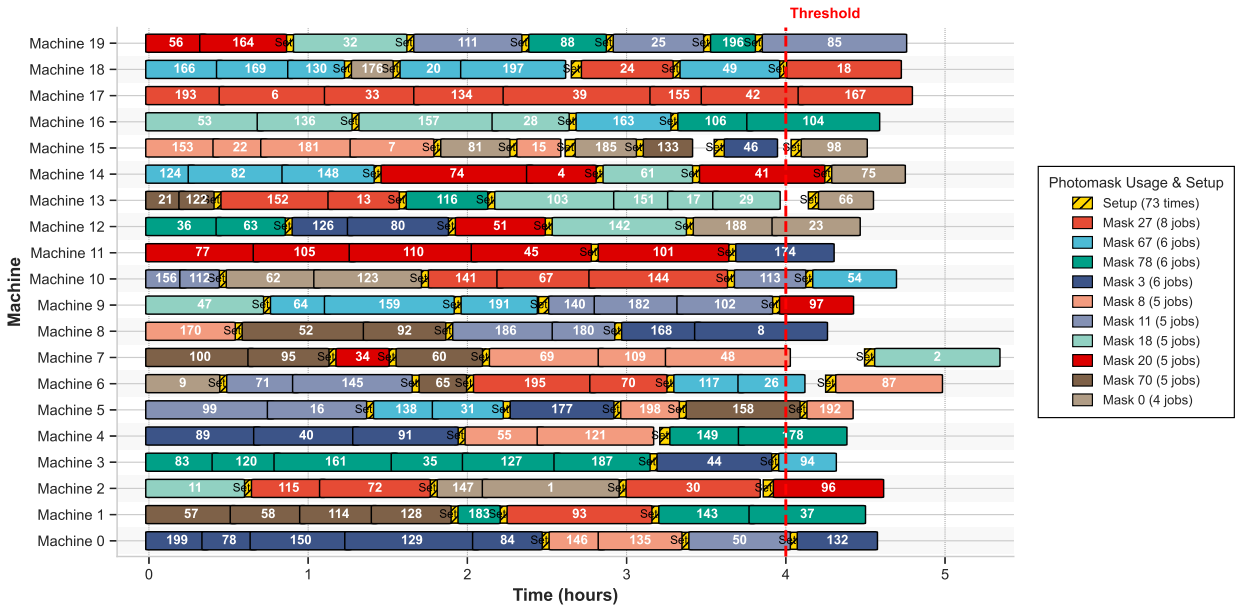


Figure 3: Gantt chart of RL-based scheduling solution for a 20-machine instance with photomask constraints and setup operations

Numerical results confirm that the proposed method outperforms three MSMH-based heuristics (MSMH-SPT, MSMH-Priority, MSMH-StepTarget), demonstrating the effectiveness of a graph-based RL approach under diverse scenarios. In addition, our framework's flexibility in handling varying resource ratios and job arrival rates suggests that it can be extended to other complex manufacturing systems.

While our current approach assumes fixed weights for the multi-objective function, real manufacturing environments often require different priority settings based on operational conditions. Future work will focus on developing a more generalized model that can adapt to various objective weight configurations without retraining, enabling flexible deployment across different manufacturing scenarios with varying performance priorities.

REFERENCES

Brody, S., U. Alon, and E. Yahav. 2021. "How Attentive Are Graph Attention Networks?". *arXiv preprint arXiv:2105.14491*.

Cai, L., J. Li, J. Wang, and S. Ji. 2021. "Line Graph Neural Networks for Link Prediction". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(9):5103–5113.

Cakici, E., and S. Mason. 2007. "Parallel Machine Scheduling Subject to Auxiliary Resource Constraints". *Production Planning and Control* 18(3):217–225.

Chang, J., D. Yu, Y. Hu, W. He, and H. Yu. 2022. "Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival". *Processes* 10(4):760.

Cho, S.-H., W.-J. Shin, J. Ahn, S. Joo, and H.-J. Kim. 2024. "Dynamic Crane Scheduling with Reinforcement Learning for a Steel Coil Warehouse". In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 16545–16552. IEEE.

Deenen, P., W. Nuijten, and A. Akcay. 2023. "Scheduling a Real-World Photolithography Area With Constraint Programming". *IEEE Transactions on Semiconductor Manufacturing* 36(4):590–598.

Ghasemi, A., R. Azzouz, G. Laipple, K. E. Kabak, and C. Heavey. 2020. "Optimizing Capacity Allocation in Semiconductor Manufacturing Photolithography Area–Case Study: Robert Bosch". *Journal of Manufacturing Systems* 54:123–137.

Huang, Y., E.-L. Hsiang, M.-Y. Deng, and S.-T. Wu. 2020. "Mini-LED, Micro-LED and OLED Displays: Present Status and Future Perspectives". *Light: Science & Applications* 9(1):105.

Hung, Y.-F., C.-H. Liang, and J. C. Chen. 2013. "Sensitivity Search for the Rescheduling of Semiconductor Photolithography Operations". *The International Journal of Advanced Manufacturing Technology* 67:73–84.

Ji, D., J. Jang, J. H. Park, D. Kim, Y. S. Rim, D. K. Hwang *et al*. 2021. "Recent Progress in the Development of Backplane Thin Film Transistors for Information Displays". *Journal of Information Display* 22(1):1–11.

Kim, E., T. Kim, D. Lee, H. Kim, S. Kim, J. Kim, *et al*. 2023. "Practical Reinforcement Learning for Adaptive Photolithography Scheduler in Mass Production". *IEEE Transactions on Semiconductor Manufacturing* 37(1):16–26.

Liu, C.-L., C.-J. Tseng, T.-H. Huang, and J.-W. Wang. 2023. "Dynamic Parallel Machine Scheduling with Deep Q-Network". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 53(11):6792–6804.

Mao, H., M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh. 2019. "Learning Scheduling Algorithms for Data Processing Clusters". In *Proceedings of the ACM Special Interest Group on Data Communication*, 270–288.

Ni, F., J. Hao, J. Lu, X. Tong, M. Yuan, J. Duan, *et al*. 2021. "A Multi-Graph Attributed Reinforcement Learning Based Optimization Algorithm for Large-Scale Hybrid Flow Shop Scheduling Problem". In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 3441–3451.

Park, J., S. Bakhtiyar, and J. Park. 2021. "ScheduleNet: Learn to Solve Multi-Agent Scheduling Problems with Reinforcement Learning". *arXiv preprint arXiv:2106.03051*.

Sun, B., H. Huang, P. Wen, M. Xu, C. Peng, L. Chen, *et al*. 2023. "Research Progress of Vertical Channel Thin Film Transistor Device". *Sensors* 23(14):6623.

Zhang, P., Y. Lv, and J. Zhang. 2018. "An Improved Imperialist Competitive Algorithm Based Photolithography Machines Scheduling". *International Journal of Production Research* 56(3):1017–1029.

ACKNOWLEDGEMENT

This work was supported by the Samsung Display Company Ltd and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00334171).

AUTHOR BIOGRAPHIES

SANG-HYUN CHO is a Ph.D. student in Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). He received a M.S. in industrial and systems engineering from KAIST. He is interested in scheduling methodologies and applications. His e-mail address is ie02002@kaist.ac.kr.

SOHYUN JEONG is a Ph.D. student in Graduate School of Data Science, Korea Advanced Institute of Science and Technology (KAIST). She received a M.S. in chemical and biomolecular engineering from KAIST. She is interested in scheduling methodologies and applications. Her e-mail address is sohyunj1224@kaist.ac.kr.

JIMIN PARK is a Ph.D. student in Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). She received a B.S. in industrial and systems engineering from KAIST. Her research interests include machine learning for scheduling and automation. Her e-mail address is jiminpark@kaist.ac.kr.

HYUN-JUNG KIM is an Associate Professor with the Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). She received B.S., M.S., and Ph.D. in industrial and systems engineering from KAIST. Her research interests include discrete event systems modeling, scheduling, and control. Her email address is hyunjungkim@kaist.ac.kr. Her website is https://msslab.kaist.ac.kr.

BOYOON CHOI has been affiliated with Samsung Display Co, and worked as a production scheduling system engineer. She received M.S. in Advanced Material engineering from Sungkyunkwan University. She is interested in reinforcement learning to improve FAB production scheduling. Her e-mail address is byc.choi@samsung.com.

PAUL HAN has been affiliated with Samsung Display Co, and worked as a production scheduling system engineer. He received M.S. in industrial engineering from Korea University. He is trying to integrate optimization techniques into FAB production scheduling. His e-mail address is pu.han@samsung.com.