Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

SEMICONDUCTOR FAB SCHEDULING WITH SELF-SUPERVISED AND REINFORCEMENT LEARNING

Pierre Tassel
Benjamin Kovács
Martin Gebser
Konstantin Schekotihin

Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
Klagenfurt am Wörthersee 9020, AUSTRIA

Patrick Stöckermann
Georg Seidel

Infineon Technologies AG
Siemensstraße 2
Villach 9500, AUSTRIA

ABSTRACT

Semiconductor manufacturing is a complex, costly process involving a long sequence of operations on limited, expensive equipment. Recent chip shortages and their impacts have highlighted the importance of semiconductors in the global supply chains and how reliant on those our daily lives are. Due to the investment cost, environmental impact, and time scale needed to build new factories, it is difficult to ramp up production when demand spikes. This work introduces a method to successfully learn to schedule a semiconductor manufacturing facility more efficiently using deep reinforcement and self-supervised learning. We propose the first adaptive scheduling approach to handle complex, continuous, stochastic, dynamic, modern semiconductor manufacturing models. Our method outperforms the traditional hierarchical dispatching strategies typically used in semiconductor manufacturing plants, substantially reducing each order's tardiness and time until completion. Consequently, our method yields a better allocation of resources in the semiconductor manufacturing process.

1 INTRODUCTION

Modern semiconductor manufacturing is one of the most complex industrial processes to date (Ravi 2021). Typical production sequences involve a multitude of complex elementary operations handled by highly specialized tools that require specific setups according to variable product and process designs (Yugma et al. 2015). Over the years, Moore's Law has pushed the economic and manufacturing limits of industry by approximately doubling the number of transistors per given area every two years. On the other hand, the costs for research and development, manufacturing, and testing become increasingly elevated with each new chip generation (Rupp and Selberherr 2010). Thus, increasing semiconductor production capacities takes time, and it is impossible to ramp up chip output overnight (Ravi 2021). In this regard, optimizing production processes is an important prerequisite for mitigating the current critical market situation. Beyond the economic and manufacturing challenges, increased production facilities have a negative ecological impact. The industry is very resource-intensive, especially in terms of electricity, land, and water consumption (Kuo et al. 2022). In 2010, the semiconductor industry accounted for 1.3–2.0% of the total US electricity consumption in the manufacturing sector (Gopalakrishnan et al. 2010). The chip-making process is also water-intensive, and an average semiconductor factory consumes around 20 000 tons of water a day with an estimate of 97.26% used during manufacturing (Frost and Hua 2017). Increasing semiconductor fab efficiency has thus been identified as one of the ways to achieve ecological benefits while reducing the environmental impact of semiconductor manufacturing (Infineon Technologies AG 2022).

Applying machine learning techniques to combinatorial optimization problems like planning and scheduling has recently emerged as a prospective research area (Bengio et al. 2021). As most of these

problems are *NP*-complete or even harder (Garey et al. 1976; Sotskov and Shakhlevich 1995), exhaustive search methods do not scale and quickly become inefficient. Applying machine learning techniques to scheduling problems has been shown to be a promising approach to generate good approximate solutions in reasonable time (Park et al. 2021; Zhang et al. 2020; Stricker et al. 2018). Recently, Tassel et al. (2022) suggested an RL-based method that can handle very large instances of the resource constraint scheduling problem. However, previous work focuses on scenarios that significantly simplify real-world problems (Stricker et al. 2018; Tassel et al. 2021). That is, considered problem instances represent finite, deterministic, and static scheduling problems with predefined sets of machines, jobs, and their operations, which remain constant during the scheduling process. Therefore, our work is the first to consider the *continuous*, *stochastic*, and *dynamic* scheduling process of a semiconductor fab, addressing a wide range of challenges encountered in large-scale production environments.

Contributions. In this paper, we propose a novel adaptive scheduling method to improve the yield and reduce customer order delays in a realistic semiconductor manufacturing environment. Using *Reinforcement Learning* (RL) and *Self-Supervised Learning* (SSL), we train a single RL agent's *Neural Network* acting as a global dispatcher. Our contributions can be summarized as follows:

RL Training Approach. Due to the specificities of the semiconductor manufacturing process and our model's architecture, we have developed a training loop using *Self-Supervised Learning* to pre-train part of our network and derivative-free *Evolution Strategies* to train another part of the network.

Feature Set. For each wafer lot, we have designed a feature set that can efficiently represent the current status of the lot in the manufacturing process.

Objective Function. We have designed an objective function to compare the quality of different schedules. This function aims to reduce the average tardiness and cycle-time, thus increasing the throughput of each lot type while incorporating their priorities. The function also considers unfinished lots (i.e., the WIP) in the computation of the score, correcting potential bias in evaluating a continuous production environment.

Model Architecture. We introduce an order- and size-invariant neural network architecture for a single-agent dispatcher using global self-attention on the features of each wafer lot to be scheduled. Our architecture can efficiently encode information such as the type of a required tool family without increasing the dimensions of the input representation.

Empirical Evaluation. We evaluate our method on a large-scale, continuous, stochastic, and dynamic semiconductor simulation model from the literature (Hassoun et al. 2020), using an open-source event-based semiconductor fabrication plant simulation (Kovacs et al. 2022). This testbed was designed to help researchers assess the performance of integrated dispatching strategies and aims to credibly reflect the complexity of modern semiconductor manufacturing. It includes two models, one representing a High-Volume/Low-Mix (*HV/LM*) production scenario and the other a Low-Volume/High-Mix (*LV/HM*) production plan. Our approach outperforms traditional dispatching heuristics in both of these scenarios, achieving so far unmatched state-of-the-art performance.

2 BACKGROUND

The scheduling problem in semiconductor manufacturing can be modeled as a variant of the Job-shop Scheduling Problem (JSP) (Gupta and Sivakumar 2006). In classical JSP, given n jobs and m machines, each job operation must be non-preemptively processed by exactly one machine within a minimal makespan, i.e., the total completion time for all jobs. Although simple in appearance, the problem is *NP*-complete, and unless $NP = P$, it cannot be solved efficiently (Applegate and Cook 1991). Due to the limited practical applicability of the classical JSP, multiple variants of this problem, which better reflect real-world scheduling scenarios, have emerged over the years (Potts and Strusevich 2009). For instance, in flow-shop scheduling, each job has exactly m operations, and an operation i is non-preemptively executed on machine i. In a flexible JSP, for every operation, there can be several machines able to process it. Complex JSP extends the flexible variant with *batching* machines, reentrant flows, sequence-dependent setup times, and release

dates (Knopp et al. 2017). The term *batching* is used in the context of parallel processing and/or in that of sequential processing (often referred to as *cascading* or *serial batching*), denoting capacities for overlapping the execution times of multiple jobs' operations (Huang and Wu 2017).

However, a realistic model of the semiconductor manufacturing process cannot be directly mapped to existing JSP variants. Therefore, (Hassoun et al. 2020) presented *SMT2020*, a set of stochastic semiconductor fab simulation models, aiming to credibly represent the complexity of modern semiconductor manufacturing. These models go beyond previously proposed semiconductor simulation testbeds such as MiniFab (Spier and Kempf 1995), MIMAC (Hassoun and Kalir 2017), Harris (Kayton et al. 1997), and SEMATECH 300mm (Kiba et al. 2009) by including larger-scale datasets and more realistic characterizations of manufacturing processes, incorporating the following features:

1. **Different Simulation Scenarios:** The simulation includes models of Low-Volume/High-Mix (*LV/HM*) and High-Volume/Low-Mix (*HV/LM*) wafer fabs, where mix stands for high/low variety of products, and volume for small/large quantities, respectively. The *LV/HM* scenario includes 10 different products with 10 associated routes, reflecting make-to-order (MTO) environments for supplying foundries, producing logic or memory devices. In the *HV/LM* setting, 2 high-volume products represent dedicated fabs, typically producing logic devices in a make-to-stock (MTS) environment.
2. **Realistic Scale:** Each product has a dedicated route with at least 500 steps, reflecting the complexity of modern wafer fabs. Similarly, both scenarios contain 106 tool groups organized in 12 families, sometimes called machine families or work centers. The *LV/HM* scenario includes a total of 1 265 machines, while *HV/LM* features 1 071 machines. At the beginning of the scheduling horizon, there are 2 700 lots in progress for the *LV/HM* dataset and 3 635 for the *HV/LM* scenario.
3. **Machine Availability:** Simulation models include scheduled downtime due to planned tool maintenance and unscheduled downtime due to machine breakdown.
4. **Batching and Cascading:** *SMT2020* scenarios involve batch and/or cascade processing for certain tool groups.
5. **Loading and Transportation:** Loading and unloading times on the tools and transportation times between the different tool groups are considered in the simulation.
6. **Machine Setup:** Tools can have operation- and sequence-dependent configurations, which must be set up on a machine before processing operations. A configuration might imply a minimum/maximum number of operations to be performed before it can/must be changed. Moreover, some operations require reconfiguring a machine, even if the same configuration was in place before.
7. **Tool Family Specificities:** Certain functional areas of a fab have specific constraints. For instance, steppers in the photolithography area have an *lot-to-lens* dedication, forcing the product to revisit the same tool in a future step of the route. A production process with some probability can skip metrology steps performing quality control. Dry etching steps have a Critical Queue Time (CQT) constraint enforcing the next operation to be performed with no more than a given amount of time delay.
8. **Lot Priorities:** Lots can have different priorities: *Super Hot Lots* have the highest and *Hot Lots* have a higher priority at all stages of processing than regular lots.
9. **Variable Lot Size:** The processing time of certain operations might depend on the number of wafers in a lot, which can vary from one lot to another.

SMT2020 provides hierarchical dispatching heuristics, combining 4 stages in the following order of significance:

1. **Critical Queue Time:** Lots with a CQT constraint in the previous operation have the top priority.
2. **Lot Priorities:** Lots must be scheduled according to their priorities, i.e., *Super Hot Lots* scheduled first, then *Hot Lots*, and finally, the regular lots.
3. **Setup Avoidance:** Reconfiguration of a tool is time-consuming; therefore, it is preferable to schedule operations in a way that avoids switching the setup.

4. **Generic Dispatching Heuristic:** To break ties, the *HV/LM* and *LV/HM* scenarios use First-Come-First-Serve (FCFS) or Critical Ratio (CR) heuristics, respectively, where CR considers the due date and the remaining processing time required to complete a lot.

The resulting simulation models are substantially different from the static and deterministic JSP, where all job and machine parameters are known in advance, and the environment is not changing during the computation and execution of a schedule. In contrast, the *SMT2020* scenarios are dynamic and stochastic as changes in the environment, like the release of new lots with different priorities or unscheduled downtimes of machines, might occur with some probability at any point in time.

3 RELATED WORK

The Semiconductor Factory Scheduling Problem (SFSP) is often decomposed into tool scheduling problems (single machine job-shop), work center scheduling (parallel machine job-shop), or work area scheduling (flexible job-shop) (Mönch et al. 2011). However, decomposition approaches merely approximate the SFSP since combinatorial optimization methods, like *Constraint Programming* (CP) or *Mixed Integer Programming* (MIP), do not scale up to the full problem due to the high implementation efforts and long computation times (Branke et al. 2016). That is, a realistic instance of SFSP contains millions of operations to schedule each year, thus going far beyond the capabilities of modern optimization algorithms (Kovács et al. 2021). On the other hand, deterministic approximations of SFSP might reduce the robustness of schedules in presence of unexpected events, like machine breakdowns or reworks. Finally, SFSP is a dynamic problem where new customer orders constantly arrive, requiring recomputing a schedule once the existing one gets inconsistent with new data.

Combined methods try to mitigate scalability issues by applying combinatorial optimization only for bottleneck machines (often lithographic tools) while using dispatching heuristics for other operations (Govind et al. 2008; Ham and Cho 2015). Often, such partial optimization approaches only marginally improve over deterministic heuristic solutions, which have in turn been shown to be less effective than meta-heuristics constructed for stochastic SFSP-like settings (Nguyen et al. 2013).

The drawbacks of combinatorial optimization methods on large and stochastic scheduling problems made dispatching heuristics a de-facto industry standard. Respective scheduling systems only manage to approximate optimal solutions, but they compute results in seconds and, therefore, can react to changes in real-time (Waschneck et al. 2016; Ham and Cho 2015). Any dispatching heuristic is a handcrafted heuristic designed by experts taking experimental studies and personal experience on typical scheduling scenarios into account. The quality of different dispatching heuristics is evaluated using SFSP simulators with various objectives like reducing mean cycle-time, average tardiness, and delayed lots. Although algorithms based on dispatching heuristics have shown good runtime results, there is no universal set of heuristics that work best for all evaluation scenarios. For instance, their performance depends on the current Work-In-Progress (WIP) (Nyhuis and Wiendahl 2012), selected key performance indicators (Uzsoy et al. 1993), and types of machines used (Fordyce et al. 2015).

Attempts to apply RL to semiconductor manufacturing aim to overcome the issues of dispatching heuristics in complex SFSP environments. Waschneck et al. (2018) propose a multi-agent RL approach in which each agent represents a work center, i.e., a tool family. This work models a factory with 4 work centers and 48 WIP lots to schedule, where transportation times and machine breakdowns are disregarded. Agents are trained in two phases: *(1)* only one agent (i.e., a work center) is trained while the others are controlled by a heuristic, and *(2)* all agents controlling the 4 work centers are trained together. Results obtained with the trained agents are comparable to the original dispatching strategy, but as the action space depends on the number of WIP lots, the approach is difficult to generalize. Stricker et al. (2018) propose a single-agent RL method for a testbed with 3 work centers and a total of 8 machines, without setup changes between different kinds of operations. The dispatching agent can select from a set of 12 actions to decide how to schedule a given lot. Since the action space depends on the number of machines, a neural

network trained on one fab cannot be reused for others and it might be difficult to apply this approach to thousands of machines as in *SMT2020*. However, the trained agent improves the initial dispatching strategy significantly by 8% better fab utilization.

4 LEARNING ALGORITHM

This work aims at the development of a scheduler for the full SFSP, in which the objective is to efficiently dispatch the lots to schedule on the machines so that metrics like mean cycle-time, average tardiness, and the number of delayed lots are minimized.

4.1 Approach Overview

Following the Markov Decision Process (MDP) formulation introduced by (Tassel et al. 2022), the state-transition function of our deep RL agent considers sequences of lots rather than a fixed number of actions assigning lots to machines. In particular, at any timepoint when a set of lots must be dispatched, the agent is trained to order lots in a sequence in which they are assigned to machines using a set-up avoidance scheme. In the following, we formulate dispatching decisions on lot operations to be scheduled in terms of an MDP, including the state space $\mathscr{S}$, action space $\mathscr{A}$, reward r_t and transition $s_{t+1} \sim p(\cdot \mid s_t, a_t)$ functions.

State Space $\mathscr{S}$. A state $s_t \in \mathscr{S}$ is given by the set $\mathscr{L}^t$ of legal lots at timepoint t. A lot l is *legal* if its operation sequence $O_l = (\tau_1, \ldots, \tau_n)$ comprises an operation τ that can be scheduled to free resources at t. Moreover, each lot l is associated with a set of features: *(1)* critical ratio, i.e., a ratio of the time remaining till the due date d_l to the expected processing time to finish the job; *(2)* remaining time until the deadline d_l; *(3)* total waiting time of the lot since its release; *(4)* waiting time of the lot since the last operation; *(5)* number of lot-to-lens dedications until lot completion, i.e., once a machine is used to perform a specific operation, it will always be used for remaining operations $\tau_i \in O_l$; *(6)* lot priority w_l; *(7)* sum of mean processing times for all remaining operations of the lot; *(8)* minimum machine setup time to perform the current operation of the lot; *(9)* mean processing time $p_\tau \in \mathbb{N}^+$ of the current operation $\tau \in O_l$; *(10)* number of available machines with compatible setup; *(11)* minimum batch size; *(12)* maximum batch size; and *(13)* the family of the required tool group.

Action Space $\mathscr{A}$. The action space $\mathscr{A}$ is composed of all legal lots at timepoint t, i.e., $\mathscr{A} = \mathscr{L}^t$. Hence, the action space is discrete, and the cardinality of $\mathscr{A}$ may vary from one timepoint to another. The agent receives as input a set of dispatchable lots for the current timepoint t, and provides as output the order in which to dispatch the given lots. For each lot $l \in \mathscr{L}^t$, the network produces a scalar representing the priority of l. The ordered set of lots to dispatch is then obtained by sorting the lots according to their priority scores.

Reward r_t. Usually, SFSP uses the average cycle-time, tardiness, and the number of delayed lots for each lot type to compute the quality of a schedule. Unfortunately, evaluations of these criteria provide meaningful results only when computed for a complete schedule and might be misleading for incomplete ones. Therefore, we have designed an objective function assigning scores for any (in)complete schedule. This function penalizes tardiness of a lot and long cycle-times while rewarding high throughputs. Moreover, the score for each lot is weighted based on its priority.

Our objective function comprises two parts: one for the lots already finished at the time of the evaluation and another for the lots still in the WIP. Both parts are then summed together to obtain the final result. Considering lots in the WIP for evaluating a schedule is particularly important in the case of a continuous scheduling environment such as SFSP, as one potential bias of a method is to only optimize metrics for the evaluated time windows but risk bringing the fab into a catastrophic state in the future.

The first part, only considering lots finished at the evaluation time t, is defined as:

$$O_f = \sum_{k_i \in K} \frac{1}{\|k_i\|} \sum_{l \in k_i} \begin{cases} w_l * (p + (c_l - d_l)) & \text{if } c_l > d_l \\ 0 & \text{otherwise} \end{cases}$$

where K is the set of all lot types, $\|k_i\|$ the number of lots of type k_i, c_l the completion date and d_l the deadline for lot l. The constant term p allows penalizing delayed lots independently of whether the tardiness is large or small. The second part, focusing on lots in the WIP at time t, is defined as:

$$O_p = \sum_{k_i \in K} \frac{1}{\|k_i\|} \sum_{l \in k_i} w_l * x_{i,l}$$

$$x_{i,l} = \begin{cases} p + (t - (d_l - (a_i * e_l))) & \text{if } d_l - (a_i * e_l) < t \\ 0 & \text{otherwise} \end{cases}$$

where a_i is the average cycle-time of lots of type k_i computed on the finished lots, and e_l the sum of mean left-to-process operations' processing time. The term $d_l - (a_i * e_l)$ evaluates the deadline for lots in the WIP by forecasting the completion date of a given lot based on the average processing time for operations to be performed until completion. Thus, the minimization of the objective function $O = O_f + O_p$ leads to the reduction of tardiness, increase in throughput, and decrease in the average cycle-time for each lot type, just as the original SFSP objective. Note that, due to the specificities of our training approach, which employs a derivative-free optimization method, we do not require a continuous and differentiable reward function.

State Transition. At each decision timepoint t, the simulator provides information about the current state s_t of a fab, and awaits an ordered list of lots to dispatch. Similar to static dispatching heuristics, the ordered list is processed using a hierarchical strategy, first giving higher priority to lots with a CQT constraint, then lots' priorities (i.e., *Hot Lots*), and finally the original order provided by the agent. The simulator then allocates the lots to their associated free machines in the given order until no free resources remain or all lots are scheduled. Next, new events from the events stack are handled and the timepoint is incremented. In *SMT2020*, lots are assigned to machines using the following heuristic strategy:

(1) If there is a lot-to-lens dedication constraint, assign to the mandatory machine.
(2) If the current operation does not require a setup, assign to a machine without setup, if any. Otherwise, assign to the machine with the highest waiting time.
(3) If the operation requires a setup, assign to a machine with this setup, if available. Otherwise, assign to the machine with the lowest setup time.

This strategy is greedy as it merely determines advantageous allocations at a timepoint t, neither backtracks nor considers future incoming lots, i.e., no "lookahead". However, since *SMT2020* supplies information about the legal lots at timepoint t only, the machine allocation strategy is reused by our agent.

4.2 Policy Architecture

The agent policy is a function $\pi_\theta(\mathscr{A} \mid s_t)$ that maps the current state of the simulator s_t to the ordered set $\mathscr{A}$ of legal actions (lots) to dispatch for the current timepoint t. π_θ is implemented as a deep neural network of the structure illustrated in Figure 1.

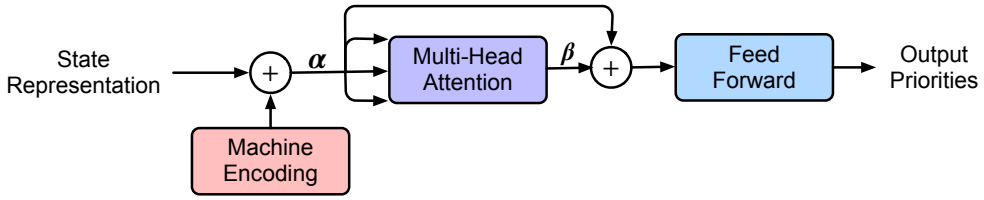

Figure 1: Policy network architecture with an embedding encoding, injecting information about the tool family required for an operation. The network outputs a priority score for each lot representation.

Before feeding the state representation to the policy neural network, we rescale the features of the legal lots using a virtual batch normalization from statistics over a 2-months horizon collected before training. For each lot representation $l \in s_t$, we inject information about the tool family required to perform the current operation, using a learned encoding, and keep the first 12 features as the lot representation (i.e., the 13-th feature is the tool family index that is only used by the learned encoding). Encodings are often used to inject positional information via absolute positional embeddings (Vaswani et al. 2017) or relative bias factors (Shaw et al. 2018), where a positional encoding is either fixed or learned (Haviv et al. 2022). In our case, the injected information is the family of the tool required to perform the current operation on a lot. We learn an associate machine encoding for each of the 12 tool families and add it to the input lot representation. Moreover, we experimented with input embedding before the machine encoding, similar to (Vaswani et al. 2017), but observed degradation in performance.

The transformed state representation s'_t is passed to a double-head self-attention mechanism and a simple, position-wise, fully connected Feed-Forward Network (FFN). We employ a residual connection with scaling factor (Szegedy et al. 2017) around s'_t and the FFN:

$$\boldsymbol{y} = \alpha \boldsymbol{x} + \beta f(\alpha \boldsymbol{x})$$

with α and β as two learnable parameters. The multi-head attention layer allows the network to determine the priority of each lot not only based on the features of the given lot but taking all lots to schedule at the current timepoint into account. It contains two attention heads, each of dimension $d_{head} = 6$. The FFN comprises three linear transformations of dimension 16 with a *tanh* activation in between:

$$\mathrm{FFN}(x) = \tanh(\tanh(xW_1 + b_1)W_2 + b_2)W_3 + b_3$$

Similar to the transformer architecture (Vaswani et al. 2017), we apply the FFN to each lot representation separately. The inner layers have dimensionality $d_{FFN} = 16$, while the network outputs a scalar representing the estimated lot priority.

4.3 Training

With an average of 7 254 125 allocated steps per year, training the policy network is challenging since the search space is highly combinatorial. Also, agents need to be trained with a long horizon since some products have a long cycle-time with completion up to three months in the future, while disruptions such as machine breakdown occur relatively rarely. Therefore, we use a training horizon of 6 months, as it provides a good balance between training runtime and the diversity of events handled by an agent.

4.3.1 Self-Supervised Learning

While the multi-head attention and FFN layers are trained together, the machine encoding is trained separately using a self-supervised pretext task (Doersch et al. 2015). The goal of the machine encoding is to inject information about the required tool family into the lot representation. This information is very important as different tool families have very different processes and constraints; for example, lithographic tools are usually considered critical due to their scarcity, etching steps often have a CQT constraint with their next operation, and other tool families also have their specificities. For the machine encoding, we propose to train a network as shown in Figure 2, having a similar architecture as the policy network. The network receives a set of lot representations as input and yields a probability distribution over all tool families as output for each lot.

Therefore, given a set of N training state representations $D = \{x_i\}_{i=0}^{N}$ sampled from a 2-months horizon simulation, the self-supervised training objective that the model must learn to solve is:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(x_i, \theta) + \lambda \|\theta_e\|^2$$

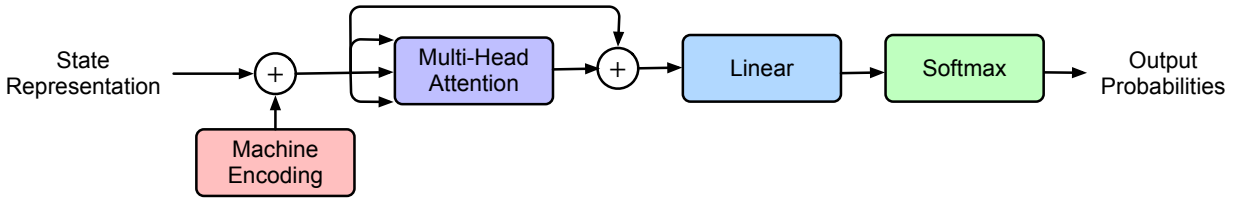

Figure 2: Self-supervised pretext task architecture. The lot representation is extended with information about the required tool family, which then must be correctly predicted by the network.

where the loss function $loss(.)$ is the cross-entropy between the predicated and the ground-truth tool family, θ the parameters of the network, θ_e the parameters of the machine encoding, and λ the regularization constant. This L_2 regularization is required to prevent the encoding from taking too large values and, thus, destroying the information about lot features. The $\lambda = 0.2$ regularization constant is tuned such that the performance on the downstream task (i.e., the fab dispatching) is maximized.

The pretext task is trained until convergence using Stochastic Gradient Descent (SGD) and backpropagation. The multi-head attention and the linear layer are then discarded, and only the learned machine encoding weights are kept frozen for the downstream task.

4.3.2 Policy Network Training

Objective functions of scheduling problems and discrete sets of possible job allocations result in learning environments with non-smooth objectives, which are unsuitable for training by backpropagating gradients. We overcome this issue by combining two methods: estimating a gradient with the Natural Evolution Strategies (NES) algorithm (Salimans et al. 2017) and a modern gradient-based optimization algorithm to update weights of the policy network.

NES belongs to a broad family of Evolution Strategies (ES), which are black-box, derivative-free methods inspired by natural evolution. When applied to the training of a policy network π_θ, NES methods use the parameter vector θ of the policy to generate a population of parameter vectors $\theta_i' \sim \mathcal{N}(\theta, \sigma^2 I)$ by adding noise sampled from an isotropic multivariate Gaussian to the original parameter vector θ, thus obtaining a set of n successor networks. Then, each of the n networks in the population is used to evaluate an objective function, the episodic reward function in our case, to estimate the *natural gradient* for the policy network:

$$\nabla_\theta F(\pi_\theta) \approx \frac{1}{\sigma * N} \sum_{i=1}^{N} F(\pi_{\theta_i'}) \varepsilon_i$$

where $\varepsilon_i \sim \mathcal{N}(0, \mathbb{I})$ is sampled from a Gaussian distribution, $\theta_i' = \theta + \sigma\varepsilon_i$, $F(\pi_{\theta_i'})$ is estimated using a single trajectory, and σ is a constant defined as $\sigma = 0.005$.

The estimated natural gradient is provided to Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-4}$, which updates the parameters θ of the policy network. Within the i-th run, we decay the learning rate with a cosine annealing as follows:

$$\eta_i = \frac{\eta_{max}}{2} \left(1 + \cos\left(\frac{i}{i_{max}} \pi \right) \right)$$

where $\eta_{max} = 0.01$ is the maximum learning rate, and $i_{max} = 40$ the maximum number of iterations.

The training algorithm continues to repeat these steps until convergence. Because NES works with function evaluations, relying on a continuous reward function is unnecessary. Therefore, we can use the previously defined unbiased and episodic objective function.

5 EMPIRICAL EVALUATION

We evaluated our approach against the original *SMT2020* hierarchical dispatching heuristics described in Section 2, using a machine equipped with AMD EPYC 7452 CPU and 128GB of RAM for benchmarking and training, with the training iterations taking 3 hours on average. As discussed in Section 3, to our knowledge, there are no CP- or MIP-based approaches in the literature for handling such large problems of stochastic, dynamic, and continuous nature. Therefore, as a baseline, we rely on proposed heuristic methods, which are the de-facto standard in the semiconductor industry (Govind et al. 2008; Ham and Cho 2015).

For each dataset, we run all dispatching strategies with a horizon of 6 months with 100 different random seeds and report the percentage of lots that meet the deadlines defined at release time as well as the cycle-time, i.e., the average number of days from release to completion per given lot type. We provide the mean and the standard deviation of each metric collected for agents trained in a considered scenario. The penalty constant was set to $p = 10$ encouraging the learner to prefer fewer delayed lots with a larger delay over a larger number of delayed lots with a very short delay each. In comparison, experimentation conducted with $p = 5$ leads to fewer lots on time but a shorter average cycle-time.

Our evaluation settings are realistic since the production scenario rarely changes during the lifetime of a fab, and Table 1 reports the aggregated metrics for each lot priority level. As we can see in Table 1(a), for the *HV/LM* dataset, our approach outperforms the state of the art, obtaining solutions of 16.68% lower cost when compared to the best-performing dispatching heuristic. Both the average tardiness and the cycle-time of lots are reduced. While the absolute reduction in solution cost is already significant, as the

Table 1: Comparison against the original dispatching heuristics on the *SMT2020* testbed for the *HV/LM* and *LV/HM* scenarios with 6 months of simulation. For each lot priority level, we report the aggregated mean and standard deviation of the percentage of lots on-time and the cycle-time. Our approach outperforms the original ones in both scenarios, especially for regular lots, representing the major part of all lots.

(a) High-Volume/Low-Mix (*HV/LM*)

Lot Type		**Ours**	**Hierarchical CR**	**Hierarchical FIFO**
Super Hot Lots	On-Time	**42.50**% ± 13.94	41.84% ± 12.45	38.06% ± 10.64
Theoretical Cycle-Time: **24.75**	Cycle-Time	**33.61** ± 0.51	33.89 ± 0.43	33.99 ± 0.44
Hot Lots	On-Time	**55.86**% ± 3.58	46.76% ± 3.77	47.66% ± 3.74
Theoretical Cycle-Time: **19.65**	Cycle-Time	**26.54** ± 0.14	26.94 ± 0.13	26.91 ± 0.12
Regular Lots	On-Time	**44.91**% ± 2.36	0.00% ± 0.00	0.00% ± 0.00
Theoretical Cycle-Time: **17.31**	Cycle-Time	**52.24** ± 1.80	70.80 ± 0.60	72.57 ± 0.61
	Cost	**33171.37** ± 1655.53	39813.37 ± 899.89	38013.67 ± 1327.73

(b) Low-Volume/High-Mix (*LV/HM*)

Lot Type		**Ours**	**Hierarchical CR**	**Hierarchical FIFO**
Super Hot Lots	On-Time	**97.38**% ± 4.63	97.19% ± 4.72	95.04% ± 5.44
Theoretical Cycle-Time: **24.75**	Cycle-Time	**33.64** ± 0.44	33.73 ± 0.42	33.86 ± 0.45
Hot Lots	On-Time	96.03% ± 3.20	**96.34**% ± 3.04	95.38% ± 3.35
Theoretical Cycle-Time: **17.31**	Cycle-Time	**23.67** ± 0.23	23.71 ± 0.23	23.80 ± 0.23
Regular Lots	On-Time	**56.47**% ± 2.36	0.00% ± 0.00	0.06% ± 0.09
Theoretical Cycle-Time: **17.31**	Cycle-Time	**44.15** ± 1.80	52.25 ± 0.30	52.38 ± 0.63
	Cost	**94349.38** ± 2716.51	108855.67 ± 2044.28	102170.71 ± 2497.89

unknown "optimal" solution (i.e., the optimal solution for each seed) has a cost greater than 0, we have an even greater relative improvement. Focusing on *Regular Lots*, the vast majority of all lots, the original dispatching heuristics are incapable of scheduling them to meet the deadline. In contrast, our approach meets the deadline for 44.91% of the lots and reduces the cycle-time by 26.21% when compared to the *CR* dispatching heuristic, thus substantially decreasing the average time from release to completion of a lot.

Aggregated results for the *LV/HM* dataset are reported in Table 1(b). As with the *HV/LM* scenario, our method outperforms the state of the art, obtaining solutions of 7.66% lower cost when compared to the best-performing dispatching heuristic. Although the metrics obtained for *Hot Lots* are very similar, our approach is always capable of reducing the cycle-time and achieving overall lower tardiness of lots. In fact, paralleling the *HV/LM* scenario, the agent vastly outperforms static dispatching heuristics when focusing on *Regular Lots*, meeting the deadline for 56.47% of them. In comparison, *Regular Lots* scheduled by standard heuristics met their deadlines in almost none of the cases. Furthermore, our approach greatly reduces tardiness and the cycle-time by about 15.5%, thus making better usage of fab resources.

Additional evaluations, including detailed results per lot type, comparison with additional dispatching heuristics from the literature, and a 1-year evaluation horizon, have been omitted due to page constraints. However, we are open to providing additional information upon request. It is worth mentioning that all scenarios that were evaluated have shown results consistent with those described above. Our approach has exhibited remarkable performance when compared to existing methods in the literature. We have observed significant enhancements across various metrics, while incurring only a minimal increase in computation time. Specifically, our approach increases the decision time per timepoint by an average of 1.02 ms.

6 CONCLUSION

We present a new scheduling method capable of improving the usage of semiconductor fab resources. Unlike prior works focusing on toy datasets with a handful of products, machines, and constraints, we aim at realistic semiconductor manufacturing models in full complexity and scale. Using a combination of self-supervised learning for an embedding encoding information about available tool families and reinforcement learning to train an attention-based model capable of considering all lots to allocate, we trained an agent to dispatch lots to machines efficiently. Extensive evaluation against a wide range of dispatching heuristics in the literature, including those proposed for the considered datasets, shows the ability of our approach to substantially improve semiconductor scheduling capacities and achieve better usage of fab resources.

While these initial results are encouraging, many challenges remain. Mainly, applying our approach in production first requires building trust of operators responsible for the production. It can be challenging to convince operators to follow such a black-box system's decisions, especially if some might initially seem counterintuitive. Another challenge is to provide a more global vision of the fab to the agent by including lots currently processed by a tool in the state representation. It would allow the agent to anticipate which lots are getting ready next and possibly decide to leave a tool free for a soon-to-be-available lot.

ACKNOWLEDGMENTS

This work was funded by KWF project 28472, cms electronics GmbH, FunderMax GmbH, Hirsch Armbänder GmbH, incubed IT GmbH, Infineon Technologies Austria AG, Isovolta AG, Kostwein Holding GmbH, and Privatstiftung Kärntner Sparkasse. We thank the anonymous reviewers for their insightful comments.

REFERENCES

Infineon Technologies AG 2022. "Industrial Applications | Making Green Energy Happen - Infineon Technologies".

Applegate, D. L., and W. J. Cook. 1991. "A Computational Study of the Job-Shop Scheduling Problem". *INFORMS Journal on Computing* 3(2):149–156.

Bengio, Y., A. Lodi, and A. Prouvost. 2021. "Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon". *European Journal Operational Research* 290(2):405–421.

Branke, J., S. Nguyen, C. W. Pickardt, and M. Zhang. 2016. "Automated Design of Production Scheduling Heuristics: A Review". *IEEE Transactions on Evolutionary Computation* 20(1):110–124.

Doersch, C., A. Gupta, and A. A. Efros. 2015. "Unsupervised Visual Representation Learning by Context Prediction". In *ICCV*, 1422–1430: IEEE Computer Society.

Fordyce, K., R. J. Milne, C.-T. Wang, and H. Zisgen. 2015. "Modeling And Integration Of Planning, Scheduling, And Equipment Configuration In Semiconductor Manufacturing. PART I. Review Of Sucesses And Opportunities". *International Journal of Industrial Engineering: Theory, Applications, and Practice* 22(5).

Frost, K., and I. Hua. 2017. *A Spatially Explicit Assessment of Water Use by the Global Semiconductor Industry*. IEEE Computer Society.

Garey, M., D. Johnson, and R. Sethi. 1976. "The Complexity of Flowshop and Jobshop Scheduling". *Mathematics of Operations Research* 1(2):117–129.

Gopalakrishnan, B., Y. Mardikar, and D. Korakakis. 2010. "Energy Analysis in Semiconductor Manufacturing". *Energy Engineering* 107(2):6–40.

Govind, N., E. W. Bullock, L. He, B. Iyer, M. Krishna, and C. S. Lockwood. 2008. "Operations Management in Automated Semiconductor Manufacturing With Integrated Targeting, Near Real-Time Scheduling, and Dispatching". *IEEE Transactions on Semiconductor Manufacturing* 21(3):363–370.

Gupta, A. K., and A. I. Sivakumar. 2006. "Job Shop Scheduling Techniques in Semiconductor Manufacturing". *The International Journal of Advanced Manufacturing Technology* 27(11):1163–1169.

Ham, A., and M. Cho. 2015. "A Practical Two-Phase Approach to Scheduling of Photolithography Production". *IEEE Transactions on Semiconductor Manufacturing* 28:367–373.

Hassoun, M., and A. Kalir. 2017. *Towards a New Simulation Testbed for Semiconductor Manufacturing*. IEEE Press.

Hassoun, M., D. Kopp, L. Mönch, and A. Kalir. 2020. *A New High-Volume/Low-Mix Simulation Testbed for Semiconductor Manufacturing*. IEEE Press.

Haviv, A., O. Ram, O. Press, P. Izsak, and O. Levy. 2022. "Transformer Language Models without Positional Encodings Still Learn Positional Information". *CoRR* abs/2203.16634.

Huang, E., and K. Wu. 2017. "Job Scheduling at Cascading Machines". *IEEE Trans Autom. Sci. Eng.* 14(4):1634–1642.

Kayton, D., T. Teyner, C. Schwartz, and R. Uzsoy. 1997. "Focusing Maintenance Improvement Efforts in a Wafer Fabrication Facility Operating under the Theory of Constraints". *Production and Inventory Management Journal; Alexandria* 38(4):51–57.

Kiba, J.-E., G. Lamiable, S. Dauzère-Pérès, and C. Yugma. 2009. *Simulation of a Full 300MM Semiconductor Manufacturing Plant with Material Handling Constraints*. Winter Simulation Conference.

Knopp, S., S. Dauzère-Pérès, and C. Yugma. 2017. "A Batch-Oblivious Approach for Complex Job-Shop Scheduling Problems". *European Journal Operational Research* 263(1):50–61.

Kovacs, Tassel, Gebser, and Sidel. 2022. "A Customizable Simulator for Artificial Intelligence Research to Schedule Semiconductor Fabrication Plants". In *ASMC*.

Kovács, B., P. Tassel, W. Kohlenbrein, P. Schrott-Kostwein, and M. Gebser. 2021. "Utilizing Constraint Optimization for Industrial Machine Workload Balancing". In *CP*, Volume 210 of *LIPIcs*, 36:1–36:17: Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Kuo, T.-C., C.-Y. Kuo, and L.-W. Chen. 2022. "Assessing Environmental Impacts of Nanoscale Semi-Conductor Manufacturing from the Life Cycle Assessment Perspective". *Resources, Conservation and Recycling* 182:106289.

Mönch, L., J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. 2011. "A Survey of Problems, Solution Techniques, and Future Challenges in Scheduling Semiconductor Manufacturing Operations". *Journal of Scheduling* 14(6):583–599.

Nguyen, S., M. Zhang, M. Johnston, and K. C. Tan. 2013. "A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem". *IEEE Transactions on Evolutionary Computation* 17(5):621–639.

Nyhuis, P., and H.-P. Wiendahl. 2012. *Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen*. Springer-Verlag.

Park, J., J. Chun, S. H. Kim, Y. Kim, and J. Park. 2021. "Learning to Schedule Job-Shop Problems: Representation and Policy Learning using Graph Neural Network and Reinforcement Learning". *International Journal of Production Research* 59(11):3360–3377.

Potts, C. N., and V. A. Strusevich. 2009. "Fifty Years of Scheduling: a Survey of Milestones". *Journal of the Operational Research Society* 60(S1).

Ravi 2021. "Chipmakers Are Ramping Up Production to Address Semiconductor Shortage. Here's Why That Takes Time".

Rupp, K., and S. Selberherr. 2010. "The Economic Limit to Moore's Law [Point of View]". *Proceding IEEE* 98(3):351–353.

Salimans, T., J. Ho, X. Chen, and I. Sutskever. 2017. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". *CoRR* abs/1703.03864.

Shaw, P., J. Uszkoreit, and A. Vaswani. 2018. "Self-Attention with Relative Position Representations". In *NAACL-HLT (2)*, 464–468: Association for Computational Linguistics.

Sotskov, Y., and N. Shakhlevich. 1995. "NP-hardness of Shop-scheduling Problems with Three Jobs". *Discrete Applied Mathematics* 59(3):237–266.

Spier, J., and K. Kempf. 1995. "Simulation of Emergent Behavior in Manufacturing Systems". In *Proceedings of SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, 90–94.

Stricker, N., A. Kuhnle, R. Sturm, and S. Friess. 2018. "Reinforcement Learning for Adaptive Order Dispatching in the Semiconductor Industry". *CIRP Annals* 67(1):511–514.

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. A. Alemi. 2017. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In *AAAI*, 4278–4284: AAAI Press.

Tassel, P., M. Gebser, and K. Schekotihin. 2021. "A Reinforcement Learning Environment For Job-Shop Scheduling". *CoRR* abs/2104.03760.

Tassel, P., B. Kovacs, M. Gebser, K. Schekotihin, W. Kohlenbrein, and P. Schrott-Kostwein. 2022. "Reinforcement Learning of Dispatching Strategies for Large-Scale Industrial Scheduling". *International Conference on Automated Planning and Scheduling*.

Uzsoy, R., L. K. Church, I. M. Ovacik, and J. Hinchman. 1993. "Performance Evaluation of Dispatching Rules for Semiconductor Testing Operations". *Journal of Electronics Manufacturing* 03(02):95–105.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. "Attention is All you Need". In *Neural Information Processing Systems*, 5998–6008.

Waschneck, B., T. Altenmüller, T. Bauernhansl, and A. Kyek. 2016. "Production Scheduling in Complex Job Shops from an Industry 4.0 Perspective: A Review and Challenges in the Semiconductor Industry". In *SAMI@iKNOW*, Volume 1793 of *CEUR Workshop Proceedings*: CEUR-WS.org.

Waschneck, B., A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek. 2018. "Deep Reinforcement Learning for Semiconductor Production Scheduling". In *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 301–306.

Yugma, C., J. Blue, S. Dauzère-Pérès, and A. Obeid. 2015. "Integration of Scheduling and Advanced Process Control in Semiconductor Manufacturing: Review and Outlook". *Journal of Scheduling* 18(2):195–205.

Zhang, C., W. Song, Z. Cao, J. Zhang, P. S. Tan, and C. Xu. 2020. "Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning". In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Neural Information Processing Systems, 1621–1633. Red Hook, NY, USA: Curran Associates Inc.

AUTHOR BIOGRAPHIES

PIERRE TASSEL is a Ph.D. candidate at the University of Klagenfurt, AUSTRIA. His research interests include utilizing reinforcement learning methods for solving combinatorial optimization problems and constraint programming. His email address is Pierre.Tassel@aau.at.

BENJAMIN KOVÁCS is a Ph.D.candidate at the University of Klagenfurt, AUSTRIA. His research interests are optimizing production processes using simulation techniques and adaptive learning methods. His email address is Benjamin.Kovacs@aau.at.

MARTIN GEBSER is professor for Production Systems at the University of Klagenfurt and Graz University of Technology, AUSTRIA. His research addresses the practical implementation and application of modern solving technology for various complex tasks, such as planning and scheduling, product configuration, and system design. His email address is Martin.Gebser@aau.at.

KONSTANTIN SCHEKOTIHIN is associate professor for Artificial Intelligence at the University of Klagenfurt, AUSTRIA. His main research topics include Artificial Intelligence for production systems and machine learning. His email address is Konstantin.Schekotihin@aau.at.

PATRICK STÖCKERMANN is a Ph.D. candidate at Infineon Technologies in Munich, GERMANY, and the University of Klagenfurt, AUSTRIA. His current research interests focus on utilizing reinforcement learning methods for semiconductor manufacturing. His email address is Patrick.Stoeckermann@infineon.com.

GEORG SEIDEL is senior manager at Infineon Technologies in AUSTRIA. Since 2000, he has been actively engaged in various industrial engineering topics, including simulation and WIP flow management. His responsibilities include overseeing the deployment of Fab Simulation across multiple Infineon sites, specifically in Kulim, Regensburg, and Villach. His email address is Georg.Seidel@infineon.com.