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.

PRODUCTIZATION OF REINFORCEMENT LEARNING IN SEMICONDUCTOR MANUFACTURING

Harel Yedidsion[1], David Norman[1], Prafulla Dawadi[1], Luke Krebs[1], Derek Adams[1], Emrah Zarifoglu[1]

[1]AI/ML team, Applied Materials, Salt Lake City, UT, USA

ABSTRACT

The semiconductor industry faces complex challenges in scheduling, demanding efficient solutions. Reinforcement Learning (RL) holds significant promise; however, transitioning it from research to a productized solution requires overcoming key challenges such as reliability, tunability, scalability, delayed rewards, ease of use for operators, and connectivity with legacy systems. This paper explores these challenges and proposes strategies for effective productization of RL in wafer scheduling as developed by Applied Materials' AI/ML team in the Applied SmartFactory® offering. In the context of this paper, *"productization"* refers specifically to the engineering and system-level integration needed to make RL solutions operational in fabs.

1 INTRODUCTION

Efficient lot scheduling in semiconductor fabs is critical but notoriously difficult to model and solve optimally, as it is an NP-Hard problem. Heuristic dispatch rules typically have tens of tunable parameters influencing Key Performance Indicators (KPIs). The dynamic nature of manufacturing necessitates frequent manual tuning. Finding the best combination for each fab condition is computationally intractable. While RL requires extensive offline training, it enables quick real-time inference, offering a potential solution. However, productizing RL for wafer scheduling involves overcoming several challenges:

Reliability: RL models must be rigorously tested for consistent performance under various conditions to prevent costly errors. We employ robust training and validation protocols, simulating diverse scenarios, and implementing fail-safes to prevent catastrophic failures (Fig. 1). **Scalability:** The large-scale nature of fabs requires managing vast state-action spaces. We address this using Graph Neural Networks (GNNs) for compression (Norman et al. 2024), along with distributed computing and GPU acceleration. Ray RLlib enables parallel training of multiple agents across environments, all updating a shared policy. **Delayed Reward**: Scheduling impacts are not immediately apparent, necessitating techniques like temporal difference learning to manage reward delays, with a focus on shorter-term problems like queue-time management (Yedidsion et al. 2022). **Ease of Use**: RL solutions should feature intuitive interfaces for operators unfamiliar with ML. We developed an intuitive UI to automate training and evaluation, visualize training processes, and incorporate continuous data updates for model accuracy (Fig. 2). Tensorboard is used to visualize the training process, tracking metrics such as reward and loss. Periodically the agent's policy is saved as checkpoints which are evaluated on unseen scenarios to further gain understanding of the RL agent's performance level and select the top-performing checkpoint. **Tunability**: RL models often function as black-box systems. It is important for RL models to feature adjustable parameters that can be customized to achieve various operational goals and accommodate changes like equipment downtime and shifts in product mix, unlike traditional black-box systems. **Connectivity with Legacy Systems**: Seamless integration with existing fab infrastructure is crucial, requiring simulators supported by historical and real-time data to train RL models effectively. We developed a connecting software layer between the RL Python code and MES/AutoSched® simulator using Applied SmartFactory® proprietary software namely Activity Manager® and Formatter® (Fig. 3).

2 CONCLUSION

Productizing RL for wafer scheduling in semiconductor fabs presents a unique set of challenges that must be addressed to realize its full potential. Integrating RL solutions into fab scheduling requires much more than simply training a single agent to solve a specific scenario. It necessitates developing a comprehensive infrastructure that automatically gathers historical and current data from Manufacturing Execution Systems (MES), trains using simulators on distributed servers, detects distribution shifts and retrains if necessary, evaluates the trained agent, offers an intuitive UI for the operator, and provides tunable outcomes by using parametrized reward weights to achieve the desired KPI tradeoff. The framework must leverage GPUs for accelerated training and inference, incorporate GNNs to manage extensive variable inputs while maintaining the structure and relationships among observation components, implement measures to prevent catastrophic failures, and address delayed rewards. The RL framework developed by the AI/ML team tackles all of these issues with the robust approach described in this paper to offer semiconductor fabs a way to harness the power of RL to transform their scheduling processes. Our system makes RL accessible to operators without requiring machine learning expertise, enhancing fab operations with tools to outperform existing scheduling techniques in terms of efficiency, cost savings, and error reduction.

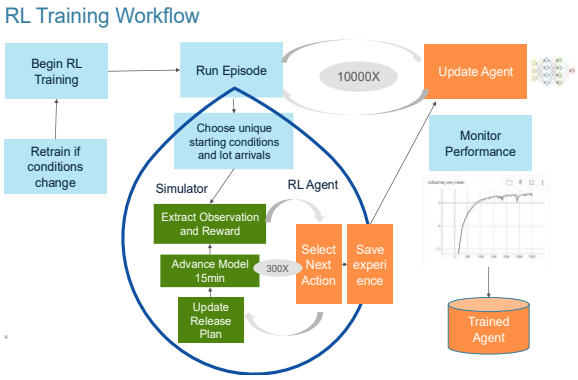

Figure 1: The RL training process, featuring blue blocks for the training framework, green blocks for the fab simulator, and orange blocks for the RL agent. There is one loop for episodes and another within each episode. Each episode has unique conditions. The observation is extracted, represented as a graph, compressed using a GNN, and passed to the agent, which selects the next action. This action is applied by the simulator, which then extracts the new observation and reward.

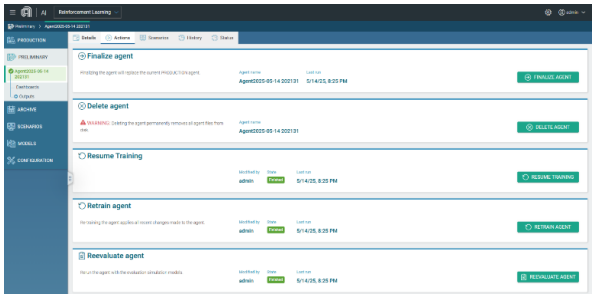

Figure 2: User Interface – we developed a user-friendly interface that allows operators to create, train and evaluate models easily. Each action button triggers background processes that take care of the tasks such as generating training scenarios, running the simulator, providing the RL agent with observations and rewards, monitoring the learning process, evaluating the agent and saving KPI results in a database.

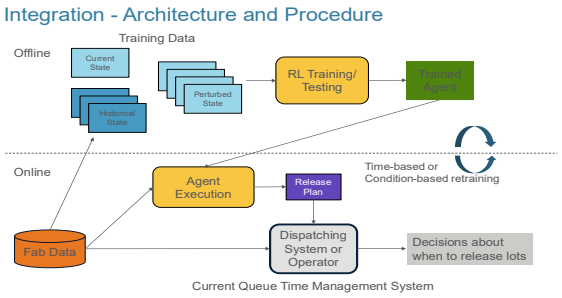

Figure 3: System Architecture –integrating the RL module with the current dispatching system,
using historic data and current data to create various training scenarios. The trained agent delivers a release plan to the dispatching system.

REFERENCES

Norman, D., Dawadi, P. and Yedidsion, H., 2024. Yield Improvement Using Deep Reinforcement Learning for Dispatch Rule Tuning. In 2024 Winter Simulation Conference (WSC) (pp. 1865-1876). IEEE.

Yedidsion, H., Dawadi, P., Norman, D., & Zarifoglu, E. 2022. Deep Reinforcement Learning for Queue-Time Management in Semiconductor Manufacturing. In 2022 Winter Simulation Conference (WSC) (pp. 3275-3284). IEEE.