Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Doctoral Consortium Track

Scalable ML Methods to Optimize KPIs in Real-World Manufacturing Processes

Benjamin Kovács
University of Klagenfurt
Benjamin.Kovacs@aau.at

Abstract

The goal of this work is to develop novel methods to solve the semiconductor fab scheduling problem. The problem can be modeled as a flexible job-shop with large instances and specific constraints related to special machine and job characteristics. To investigate the problem, we develop a tool to simulate small to large-scale instances of the problem. Using the simulator, we aim to develop new dispatching strategies using genetic programming and reinforcement learning.

1 Introduction

Efficient scheduling of semiconductor fabs remains challenging, as the problem is NP-hard, and real-world problem instances tend to be intractable for several methods due to the complexity and the size of the problem. [Bureau *et al.*, 2006] The manufacturing process usually consists of several hundred steps, while machines can be allocated dynamically, and a factory manufactures several hundreds of product types each involving different routes. The process is also subject to constraints like time coupling and machine dedications. Scheduling preventive maintenance, unexpected downtimes and batch processing also enhance the complexity.

Developing flexible and adaptive schedulers is important to maximize the exploitation of the fabs' resources and overcome supply shortages caused by the unpredictable demand.

Machine learning-based methods demonstrated the capability of tackling large-scale problems that no alternative algorithms managed before. Novel approaches like reinforcement learning (RL) outperformed previous algorithms in playing board and video games and even reached superhuman performance. Recently, solving combinatorial optimization problems (COPs) with RL was also attempted by multiple researchers [Lee *et al.*, 2019; Li *et al.*, 2020; Tassel *et al.*, 2021; Paeng *et al.*, 2021; Kuhnle *et al.*, 2021], aiming to replace approximation and heuristic-based methods used at large-scale problems. However, there are still many open problems in the field since the algorithms need to be reliable while handling dynamic settings with many constraints and uncertainty for practical applicability. For example, trustability is essential in case of real-world applications, as suboptimal decisions could lead to billions of dollars of losses and significant production delays.

This project focuses on improving key performance indicators on large-scale real-world fab scheduling problem instances. We work together with industrial partners and pay particular attention to practical applicability (including reliability and trustability) of the developed methods.

2 Methods

The project is organized in two major parts. The first part focuses on developing a toolbox to evaluate the proposed and the reference methods. For this purpose, a scalable discrete event simulator was developed with a framework to interface and monitor the dispatching agents. The second part of the project is the development and evaluation of new and improved methods to solve the introduced problem.

2.1 Simulation

An extensible simulator – including a reinforcement learning interface, and a general interface for other methods – was built in Python. The SMT2020 dataset [Kopp *et al.*, 2020] was considered as a baseline model. It consists of four large-scale fab models incorporating the challenges of modern fab scheduling like batching machines, re-entrant flow, machine dedications and uncertainties related to processing time and machine availability. The main advantages of the simulator compared to the widely used commercial solutions are the following: (a) Both the software and the base datasets are *freely available*, supporting open research and making it possible to measure progress by comparing against other methods. Additionally, there are no external library dependencies in the core code. (b) *Performance* was considered important from the beginning due to the high sample complexity of many ML-based methods. The core simulation works with minimal overhead while plugins make it easy to collect custom data required for the implemented algorithms. (c) Using the *general interface*, both planning and dispatching-oriented methods can be integrated. The *reinforcement learning* framework makes it possible creating custom *gym* [Brockman *et al.*, 2016] environments and training agents in minutes. (d) Easy evaluation using the bundled *monitoring plugins*: charts to visualize agent performance, timelines to show schedules.

Lot Type	Average cycle time	Throughput	Percentage on-time
Hot Lot 3	33 (32)	522 (499)	54 (91)
Hot Lot 4	18 (18)	523 (508)	66 (90)
Lot 3	68 (51)	19607 (18914)	17 (91)
Lot 4	39 (28)	19961 (19577)	13 (90)
Super Hot Lot	33 (N/A)	40 (N/A)	46 (N/A)

Table 1: Lot metrics for HV/LM dataset

2.2 Methods

To verify the simulator, the dispatching strategies described in the documentation dataset have been implemented: a hierarchical priority rule considering setup avoidance, the priority of lots and the critical ratio.

After the implementation of the simulator, my current focus is to develop and test AI-based methods to improve key performance indicators like the throughput of the factory and the timeliness of the orders. Key success factors for the algorithms are scalability and – considering the targeted practical application – also explainability.

Deep reinforcement learning proved to be successful in many sequential decision making problems. Therefore, we aim to analyze its potential for our complex scheduling problem. The two major challenges we face are the development of a scalable state-action representation and solving the credit assignment problem, so the agent can connect an action with the effect several steps later. Both single and multi-agent approaches are to be considered, initially focusing on a subset of the workstations (bottlenecks), while controlling the rest of the fab with alternative dispatching strategies.

The second investigated approach is genetic programming which has practical advantages compared to RL. First, instead of a continuous stepwise reward, only a single cost has to be defined for an episode, which can correspond to a combination of the KPI's values. Second, the outputted programs can be evaluated or even further refined by human experts, improving trustability. Finally, previous domain knowledge information can be incorporated by adding handcrafted strategies to the initial population.

Finally, planning-based methods are also investigated. We are analyzing the maximum size of a time window that one can handle using constraint programming or integer linear programming solvers. However, handling uncertainty related to processing times, breakdowns, and rework required frequent modification of the models and re-planning. Additionally, on toy instances, exact methods can compute optimal solutions, making possible to measure the performance of the developed methods.

2.3 Results

Using the dispatching rules, the experimental results from the dataset can be reproduced with high accuracy, verifying the correctness of the simulator implementation. 2-year simulation results of the first SMT2020 dataset are presented in Table 1, with reference values (in brackets).

The new methods are currently intensively developed, therefore only initial findings can be discussed. The genetic programming based solution – a new machine group-based heterogeneous dispatching strategy – improved the timeliness of orders compared to the handcrafted methods.

For the RL-based method, a large set of experiments is currently being prepared, with initial results expected in the coming month.

3 Future Work

In the coming months the aim is to continue improving and fine-tuning the introduced methods. The results are expected to be submitted to a conference by the summer. Meanwhile, the simulator will be released to the research community on *GitHub*, after a presentation at a conference where the paper is currently in the second stage of the review process.

We also expect to improve the collaboration with our industrial partner to obtain more real-world datasets to analyze how efficiently can agents transfer knowledge between different problem setups.

References

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[Bureau *et al.*, 2006] Mickaël Bureau, Stéphane Dauzère-Pérès, and Yazid Mati. Scheduling challenges and approaches in semiconductor manufacturing. *IFAC Proceedings Volumes*, 39(3):739–744, 2006. 12th IFAC Symposium on Information Control Problems in Manufacturing.

[Kopp *et al.*, 2020] Denny Kopp, Michael Hassoun, Adar Kalir, and Lars Mönch. Smt2020—a semiconductor manufacturing testbed. *IEEE Transactions on Semiconductor Manufacturing*, 33(4):522–531, 2020.

[Kuhnle *et al.*, 2021] Andreas Kuhnle, Jan-Philipp Kaiser, Felix Theiß, Nicole Stricker, and Gisela Lanza. Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing*, 32(3):855–876, Mar 2021.

[Lee *et al.*, 2019] Won-Jun Lee, Byung-Hee Kim, Keyhoon Ko, and Hayong Shin. Simulation based multi-objective fab scheduling by using reinforcement learning. In *Winter Simulation Conference Proceedings*, pages 2236–2247, 2019.

[Li *et al.*, 2020] Yuanyuan Li, Edoardo Fadda, Daniele Manerba, Roberto Tadei, and Olivier Terzo. Reinforcement learning algorithms for online single-machine scheduling. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 277–283, 2020.

[Paeng *et al.*, 2021] Bohyung Paeng, In-Beom Park, and Jonghun Park. Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups. *IEEE Access*, 9:101390–101401, 2021.

[Tassel *et al.*, 2021] Pierre Tassel, Martin Gebser, and Konstantin Schekotihin. A reinforcement learning environment for job-shop scheduling. *CoRR*, abs/2104.03760, 2021.