Hybrid ASP-based multi-objective scheduling of semiconductor manufacturing processes (Extended abstract)[⋆]

Mohammed M. S. El-Kholany[1,2], Ramsha Ali[1], and Martin Gebser[1,3]

[1] University of Klagenfurt, Austria
[2] Cairo University, Egypt
[3] Graz University of Technology, Austria

Semiconductor manufacturing scheduling is intricate due to the vast diversity of products, operations, and advanced machines [1]. The goal is to allocate tasks efficiently, ensuring timely delivery while meeting increasing demands. Modern wafer fabrication facilities have complex process flows, with each route undergoing hundreds of operations by distinct tool groups [2]. The environment varies from conventional setups to re-entrant flows where tasks revisit the same tools, leading to potential competition for machine access. To address these challenges, the industry often employs custom or machine-learned [3] rules for dispatching. We significantly extend our preliminary approach introduced in [4] and incorporate crucial features of realistic semiconductor fabs, including flexible machine processing, setup, batching and maintenance operations, as well as multiple optimization objectives reflecting the factory throughput, setup and batching criteria. This paper advances the scheduling approach by modeling the production process using Answer Set Programming with difference logic [5, 6], offering a comprehensive solution that considers various critical aspects of semiconductor manufacturing.

Manufacturing Scenario: We consider a *Semiconductor Manufacturing Scheduling Problem* (SMSP) inspired by the SMT2020 simulation scenario.

Hybrid ASP Approach: In our experiments using the SMSP encoding prototype [4], we observed that predetermined machine operation assignments can compromise optimal results. On the other hand, allowing complete flexibility can result in a surplus of ground rules, which can decelerate the optimization, especially when there are numerous machines in a tool group. To strike a balance, this study presents a constant named *sub_size*. This dictates the number of machines that can be designated for an operation. Specifically, a *sub_size* of 0 offers full flexibility, a value of 1 locks in the machine assignment, while any value above one restricts assignments to a *subgroup* within a tool group, encompassing up to *sub_size* machines.

Another strategy determines the subgroup to which an operation is allocated, based on a lexicographical index for operations to be processed by machines in the same tool group. The allocation is influenced by the new constant *lot_step*. When *lot_step* is set to

⋆ This work was partially 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.

0, all operations from the same lot share a common index. If it is set to 1, operations receive consecutive indexes. The reasoning behind these strategies is to ensure operations from the same lot follow each other, eliminating competition for machine access.

Within each subgroup, a setup strategy is introduced, which comes into play when the constant *by_setup* is not set to 0. The objective is to sequence setups based on the cumulative processing times of their operations, prioritizing those with longer processing duration. Subsequently, to adhere this sequence, for assigning setups and their corresponding operations to specific machines, always the least occupied machine for the upcoming setup allocation is selected. The comprehensive encoding can be accessed online. [4]

Multi-objective Optimization: Our approach to multi-objective optimization integrates minimization based on difference logic variable values [4, 7] with traditional ASP optimization capacities, using multi-shot solving functionalities [8]. This methodology is applied to set constraints on operation completion times, factoring in processing times, setup times, and maintenance.

The algorithm considers processing times for the first operations in production routes and additional times for setups when necessary. It propagates time constraints throughout the production route and the processing order of operations on machines, taking into account maintenance and setup times. For batches, the algorithm ensures synchronization of operation completion times, depending on the operation that finishes last in a batch. The approach also involves makespan minimization. The algorithm iteratively seeks to find schedules with shorter makespans until it identifies an optimal makespan. After the optimal makespan is identified, the algorithm then focuses on minimizing setup and batch violations using native ASP optimization. Setup violations are given higher priority over batch size violations due to the additional time and effort required to set up machines. In essence, the described approach balances multiple objectives, such as optimizing makespan and minimizing setup and batch violations, in a sophisticated scheduling context.

Experiments: We designed a benchmark set centered on 10 production operations for two product types from the SMT2020 scenarios [2]. These operations are processed by machines from three tool groups, encompassing re-entrant flow, batching, setups, and maintenance. In our focus, we utilized instances with 9 machines (3 per group) and an increasing lot count. Testing was conducted using clingo[DL] (version 1.4.0) on a Dell Latitude 5590, with two time limits per run: 450 seconds for makespan minimization and 150 seconds for setup/batch violations. The results reveal the impact of flexibility and strategy on the performance, highlighting challenges with scalability. Our benchmark, though extensive, only captures a fraction of larger semiconductor factories. The complexity stems from intricate setup and maintenance operations. We believe that similar scalability challenges might arise with other constraint programming encoding, but ASP with difference logic aids in faster prototyping. Our goal is to find strategies to break down large instances into manageable pieces, targeting specific bottlenecks or operation flows.

[4] `https://github.com/prosysscience/FJSP-SMT2020`

References

1. Upasani, A., Uzsoy, R., Sourirajan, K.: A problem reduction approach for scheduling semiconductor wafer fabrication facilities. IEEE Transactions on Semiconductor Manufacturing 19(2), 216–225 (2006). https://doi.org/10.1109/TSM.2006.873510
2. Kopp, D., Hassoun, M., Kalir, A., Mönch, L.: SMT2020—A semiconductor manufacturing testbed. IEEE Transactions on Semiconductor Manufacturing 33(4), 522–531 (2020). https://doi.org/10.1109/TSM.2020.3001933
3. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A.: Deep reinforcement learning for semiconductor production scheduling. In: Proceedings of the Twenty-ninth Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC'18). pp. 301–306. IEEE (2018). https://doi.org/10.1109/ASMC.2018.8373191
4. Ali, R., El-Kholany, M., Gebser, M.: Flexible job-shop scheduling for semiconductor manufacturing with hybrid answer set programming (application paper). In: Proceedings of the Twenty-fifth International Symposium on Practical Aspects of Declarative Languages (PADL'23). pp. 85–95. Springer (2023).
5. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16). pp. 2:1–2:15. Schloss Dagstuhl (2016). https://doi.org/10.4230/OASIcs.ICLP.2016.2
6. Janhunen, T., Kaminski, R., Ostrowski, M., Schellhorn, S., Wanko, P., Schaub, T.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming 17(5-6), 872–888 (2017). https://doi.org/10.1017/S1471068417000242
7. El-Kholany, M., Gebser, M., Schekotihin, K.: Problem decomposition and multi-shot ASP solving for job-shop scheduling. Theory and Practice of Logic Programming 22(4), 623–639 (2022). https://doi.org/10.1017/S1471068422000217
8. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming 19(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054