Flexible Job-shop Scheduling for Semiconductor Manufacturing with Hybrid ASP — Extended Abstract[⋆]

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

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

Classical and flexible job-shop scheduling [9, 1] constitute well-known combinatorial optimization domains, on which various local and exact search methods have been investigated (see [2] for an overview). In contrast to typical benchmark scenarios that deal with some hundreds to thousands of operations, grouped into sequences called jobs of modest length (e.g., 20 operations to be performed per job), industrial settings in semiconductor manufacturing [3, 7] involve hundreds of operations per job, taking several months to completion by a fab that performs thousands of operations each day.

Given the complexity of operations, realistic semiconductor manufacturing processes cannot be planned without uncertainties and stochastic fluctuations, such as varying process durations, occasional machine disruptions and reworking steps. Hence, simulation methods [7, 8] modeling the physical production provide indispensable means to experiment with scheduling strategies and empirically assess their expected average-case performance. The strategies applied in practice aim to optimize performance indicators, such as makespan or tardiness, locally by applying preconfigured dispatching rules [5], machine-learned decision making policies [10] or exact optimization methods for the allocation of particular machines [4]. In our work, we focus on the *SMT2020* semiconductor manufacturing simulation scenario [7] and model specific conditions like machine setups and maintenance in the hybrid framework of *ASP modulo difference logic* [6]. While long-term, global optimization of production routes with hundreds of operations is prohibitive in terms of problem size and the need to adapt to unpredictable stochastic events, our work lays the basis for the future integration of more informed exact scheduling techniques with simulation and reactive decision making methods.

Application Scenario: The SMT2020 simulation scenario specifies two kinds of semiconductor fab settings, one denoted High-Volume/Low-Mix with two types of products and the other Low-Volume/High-Mix with ten types of products. Both have in common that, depending on the product type, the production route of each lot includes a large number of operations, i.e., between 300 and 600 of them. Formally, a problem instance consists of a set $M = M_1 \cup \cdots \cup M_m$ of *machines*, belonging to *tool groups* $M_1, \ldots, M_m$, and a set $J = \{j_1, \ldots, j_n\}$ of *jobs*, where each $j \in J$ is a sequence $\langle o_{1_j}, \ldots, o_{l_j} \rangle$ of *operations*. Each operation o_{i_j} is associated to a tool group $M_{i_j} \in \{M_1, \ldots, M_m\}$, a *setup* s_{i_j} and a *processing time* p_{i_j}. The machines in a tool group M_i may undergo periodic *maintenance* procedures, subject to lower and upper

⋆ 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.

bounds on the number of processed *lots* or processing *time* after which the maintenance procedure must be repeated and occupies a machine for some period of time.

A *schedule* allocates each operation to a machine in the associated tool group, determines an order of performing the operations per machine, and also incorporates any required maintenance procedures. The resulting processing, setup and maintenance times along with waiting dependencies on predecessor operations in jobs lead to a completion time c_j for each job j. This yields the *makespan* $\max\{c_j \mid j \in J\}$, i.e., the latest completion time over all jobs, and we take its minimization as optimization objective.

Hybrid ASP Approach: As customary in ASP, we represent a problem instance in terms of specific facts and a general ASP modulo difference logic encoding, taking advantage of the clingo[DL] language and system [6]. The operations to schedule are represented by $op((L,P,R,S),G,T)$ atoms, where the tuple (L,P,R,S) identifies an operation by its lot L, product P, operation number R and setup S. In addition, G and T provide the associated tool group as well as the processing time. We adopted two strategies for machine allocation: 1) a fixed machine assignment based on the lexicographic order of operations; 2) flexible machine assignment, i.e., operations can be freely assigned to any machine in their associated tool groups. The following rules with difference logic constraints are then used to determine the start (and completion) time of each operation:

$$\begin{array}{l}
\&\mathit{diff}\{0 - O\} <= -T :- \mathit{setuptime}(O,T).\\
\&\mathit{diff}\{O - O'\} <= -T :- \mathit{later}(O,O',T).\\
\&\mathit{diff}\{O - O'\} <= -T\ -T' :- \mathit{order}(O,O',T)\ , \mathit{pmsetuptime}(O',T').\\
\&\mathit{diff}\{(L,P,R,S) - \mathit{makespan}\} <= -T :- \mathit{op}((L,P,R,S),G,T)\ ,\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{not}\ \mathit{op}((L,P,R+1,\_),\_,\_).
\end{array}$$

The first rule expresses that the required setup time (possibly zero) constitutes the earliest possible start time of an operation O. Waiting dependencies on predecessor operations in the same job or to be performed before O' on the same machine are propagated by the second rule. The third rule additionally incorporates the time T' for setup and maintenance procedures required in-between performing O and O' on the same machine. Finally, the completion time of the last operation per job is asserted as lower bound on the value of variable $makespan$, which can then be minimized by clingo[DL].

Experiments: We performed preliminary experiments on instances extracted from the SMT2020 simulation scenario. Our eleven test cases incorporate between 10 and 45 operations and from 2 to 6 machines.[4] While such instances may appear to be small, we observed a sharp transition from trivial runs to aborts at 600 seconds when the number of operations increases. Also in some cases, the higher combinatorics outweigh the advantages of flexible over fixed machine assignment regarding the feasible makespan.

Future Work: Our preliminary results on the use of ASP modulo difference logic for scheduling operations in semiconductor manufacturing reflect work in progress. As future work, we target the improvement of memory and search efficiency, addition of yet missing capabilities like batch processing and pipelining of operations, decomposition techniques making better tradeoffs than either fixed or fully flexible machine assignment, and favorable integration with simulation and reactive decision making methods.

[4] Instances are available at: `https://github.com/prosysscience/FJSP-SMT2020`

References

1. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
2. Chaudhry, I., Khan, A.: A research survey: Review of flexible job shop scheduling techniques. International Transactions in Operational Research **23**(3), 551–591 (2015)
3. Da Col, G., Teppan, E.: Industrial size job shop scheduling tackled by present day CP solvers. In: Proceedings of the Twenty-fifth International Conference on Principles and Practice of Constraint Programming (CP'19), pp. 144–160. Springer-Verlag (2019)
4. Forstner, H.: OptiGen scheduling and optimization (2022), `https://media.infineon.com/channel/89133/corporate/1/13q7sMjwwJVAwRCdq2a2mj`
5. Holthaus, O.: Efficient dispatching rules for scheduling in a job shop. International Journal of Production Economics **48**(1), 87–105 (1997)
6. Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., Wanko, P.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming **17**(5-6), 872–888 (2017)
7. Kopp, D., Hassoun, M., Kalir, A., Mönch, L.: SMT2020—a semiconductor manufacturing testbed. IEEE Transactions on Semiconductor Manufacturing **33**(4), 522–531 (2020)
8. Kovács, B., Tassel, P., Ali, R., El-Kholany, M., Gebser, M., Seidel, G.: A customizable simulator for artificial intelligence research to schedule semiconductor fabs. In: Proceedings of the Thirty-third Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC'22), pp. 106–111. IEEE Computer Society Press (2022)
9. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research **64**(2), 278–285 (1993)
10. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A.: Optimization of global production scheduling with deep reinforcement learning. Procedia CIRP **72**, 1264–1269 (2018)