Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

AUTONOMOUS SCHEDULING IN SEMICONDUCTOR BACK-END MANUFACTURING

Jelle Adan
Alp Akcay

School of Industrial Engineering
Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

John Fowler

Department of Supply Chain Management
Arizona State University
PO Box 874706
Tempe, Arizona 85287-4706 USA

Marc Albers

School of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

Michael Geurtsen

Industrial Technology and Engineering Centre
Nexperia
Jonkerbosplein 52
Nijmegen, 6534AB, THE NETHERLANDS

ABSTRACT

Production scheduling decisions have a large impact on efficiency and output, especially in complex environments such as those with sequence- and machine-dependent setup times. In practice, these scheduling problems are usually solved for a fixed time ahead. In semiconductor back-end manufacturing, given the dynamics of the environment, it is commonly observed that a schedule is no longer optimal soon after it is made. Here, we propose time-based rescheduling heuristics that can mitigate the effect of these deviations from the schedules. We build a simulation model to represent the dynamics of the shop floor as well as its interaction with the upper management level that decides how orders are released. The simulation model, which is built and validated using real-world data, enables us to evaluate the performance of the rescheduling heuristics. By comparing the results to the case without rescheduling, it is shown that rescheduling can significantly improve relevant performance measures.

1 INTRODUCTION

Electronic devices require integrated circuits in order to operate, communicate and interact with their environment. All of these integrated circuits are produced in the semiconductor industry. The semiconductor market has increased to an annual worth of over 500 billion USD and is expected to reach an annual worth over 720 billion USD by 2027 (Fortune 2020). The market will continue to grow, driven by factors as increasing number of data centers, the use of advanced technologies and the growing popularity of Internet of Things (IoT) devices and smart homes (Albers 2022). In order to keep up with the high demand, semiconductor manufacturers need to optimize their production schedules in order to reach the maximum output of their facilities.

An overview of the production scheduling problems found in semiconductor manufacturing systems has been provided by Mönch et al. (2011). One of these problems is unrelated parallel machine scheduling with sequence- and machine-dependent setups; we refer the reader to Chen et al. (2022) for a recent overview of the related literature. A common approach in this literature is to focus on ideal environments

with a fixed number of jobs, each having known parameters and constant machine capacities with no unexpected events or machine breakdowns. However, in reality, a production schedule which is generated under such assumptions may be inefficient or even infeasible to follow and may require some adjustments in response to deviations from the original production schedule. Advances in data and artificial intelligence technologies create the potential to track the real-time status of the shop floor and automate such adjustments to adaptively control the production activities, leading to what we refer to as *autonomous scheduling*. It is of key importance to be able to quantify the potential gain from such an autonomous scheduling setting, which will eventually use real-time data to make rescheduling decisions itself.

In this paper, we focus on back-end semiconductor manufacturing (i.e., making the final products from the wafers received from the front-end) and assume the availability of an existing algorithm to solve the unrelated parallel machine scheduling problem with sequence- and machine-dependent setups. This algorithm sends production schedules to the shop floor in periodic intervals. Motivated from real-life practice of Nexperia, a global semiconductor manufacturer producing more than 90 billion products annually (Geurtsen et al. 2022), we consider that the adjustment on a schedule can be made via a so-called rescheduling action. At the rescheduling moment, the scheduling algorithm is run with the new information (i.e., obtained via real-time monitoring of the shop floor). However, it is unknown in practice when to do the rescheduling. In addition, it is necessary for an effective rescheduling policy to have the ability to influence the amount of orders that are released to the shop floor (so that the available production capacity can be used as much as possible without introducing extra work in process inventory).

The objective of our paper is to address the following research questions: (1) How to represent the dynamics in the shop floor and its interaction with the upper level that controls the order release to the shop floor? (2) What is the potential benefit that can be obtained in real-life practice by using the rescheduling actions? We address the first question by building a simulation model that is calibrated with real-life data and equipped with a feed-back loop architecture that passes the current status of production (i.e., the status of equipment and jobs) to the so-called job-generator module in the simulation model. We use this simulation model to answer the second research question. In particular, we propose a simple rescheduling rule that triggers the running of the scheduling algorithm based on the time passed since the last schedule generation, frequently referred to as time-based or periodic rescheduling (Liu et al. 2016). We test the performance of this policy by using our simulation model and quantify the effects on the key performance measures such as tardiness, production quantity, the idle time of the equipment, and the number of rescheduled jobs.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of the link between several layers of decision making in a typical back-end production setting. In Section 3, we provide the details of our simulation model with an emphasis on its architecture that links shop floor execution to how orders are released to shop floor. We also explain how the simulation model is validated by using real-life data from Nexperia. In Section 4, we present our computational results. Finally, Section 5 concludes the paper with some recommendations for future work.

2 BACK-END PLANNING AND CONTROL

At large enterprises, supply chain management is commonly decomposed into multiple hierarchical management levels (Steven 2004). Typically, the highest level aggregates production facilities, products and time periods, whereas the lowest level focuses on a particular production facility and product, on a much smaller timescale. This work focuses on the bottom three levels, i.e., from top to bottom: (i) the enterprise-wide tactical level, (ii) the operational management level and (iii) manufacturing execution.

The tactical level comprises the bulk of production planning. Based on demand forecasts, its main task is to match supply and demand over several months up to a year. It oversees the entire chain: from external suppliers, through front-end and back-end production facilities, to customers. However, the level of detail is limited and rough-cut models are used to provide quick feasibility checks. On a regular basis, usually weekly, the tactical management releases orders to the back-end facilities. According to the rough-cut models, these orders supposedly fill the facility's production capacity up to the next order release.

Operational management is on the receiving side of this order release. The main task of the operational layer is to plan and control the manufacturing process, on the daily to weekly timeframe. Contrary to the tactical level, the operational management level has the complete overview of the available equipment and other manufacturing resources. With this information, it prepares a schedule. This schedule has a horizon roughly up to the next order release. Since the tactical level makes use of rough-cut models, the actual horizon may lie before or after this point.

Subsequently, this schedule is communicated to the manufacturing execution layer, i.e., the shop floor. Although the shop floor is instructed to adhere to the schedule, the production area is a highly stochastic environment. Under normal circumstances the schedule is adhered to, however, uncertain events (e.g. machine breakdowns) may necessitate to deviate from the original schedule. Mostly, such necessary adjustments are made ad hoc and rarely ever communicated back to the operational management layer.

3 MODELING APPROACH

A simulation model is developed to mimic the flow of jobs from the tactical management layer all the way to the finished products. In this section, we present the details of this modeling approach. Specifically, the data acquisition is described first. Then, the simulation model, which comprises four main components (a job generator, scheduler, shop floor, and rescheduler) will be explained in detail. At last, several settings are specified and an example experiment is conducted to illustrate and verify the behavior of the simulation model.

3.1 Data Acquisition

To develop a simulation model that imitates the flow of jobs through the supply chain, as described previously, historical data of these jobs is required. This work focuses on a specific part of the back-end manufacturing at Nexperia. Particularly, the focus is on 35 machines that process a certain group of products. These machines are of different ages and use different technologies, i.e., using standard scheduling terminology, they are unrelated. Furthermore, the group of products processed on these machines consists of roughly 600 unique products. For more information about the back-end manufacturing process, the reader is referred to (Deenen et al. 2020).

Over a period of about one year, a data set that contains the past realizations of the jobs processed on these machines was constructed. Each job in this data set contains a unique identifier, a product type, as well as several other product level characteristics. This data can be obtained relatively easily from the manufacturing execution system (MES). However, to gain insight regarding the production of a particular job, another data source needs to be consulted.

In 2001, Nexperia introduced its so-called advanced warning and data collection system, abbreviated AWACS. This information system is directly connected to the equipment, and it records events into event log files. Each log entry starts with a timestamp and is followed by an event description. Events include equipment state changes, product handles and a wide range of other actions such as the start of a job or scanning a new batch of raw material (e.g. a new wafer or lead frame). To gain insight into the behavior of the equipment, these event logs are the most reliable and accurate data source available. It is a reliable source because these events are triggered automatically and do not rely on any manual actions of an operator. For example, when a sensor detects that a certain limit is exceeded, the equipment state automatically changes to the error state. Although an operator is required to resolve the error, the system does not rely on an operator to record the occurrence of the error; this happens in a fully automatic fashion. Furthermore, the system is highly accurate as it records events with a nanosecond precision.

For the purpose of this study, three quantities are derived from AWACS, for each of the historic job realizations. The simplest of the three is the product quantity of a job. The second value that is derived from the event logs is the average process speed in the production state. Some products may be processed at lower speeds compared to others (e.g. due to quality related restrictions), whereas some machines process

faster or slower in general (e.g. newer versus older machines). The third measure that is derived is the fraction of time that the machine is in the non-production state. Although the machine can be in various states, in this context it is only relevant to consider the production state (i.e., machine is processing) and the non-production state. The latter is an aggregate of all the states wherein the machine is not processing, regardless the variety of possible reasons (e.g. an error occurred or it is waiting for a raw material). Thus, this third measure is an indication of the amount of disturbances that occurred during processing of a particular job.

In summary, the data set contains a total of 5500 historical realizations of jobs. Each job has a unique identifier, and is further described by various product level characteristics and three measures that capture the details regarding its production process. How this data is used within the simulation model is covered in the next section.

3.2 Simulation Model

The entire simulation model is programmed in C# 6.0 and makes use of the CSSL library to ease implementation (Adan, Deenen, and Geurtsen 2020). The simulation model constitutes multiple components that interact with one another to produce an accurate imitation of the real-world environment, i.e., each element functions as an individual entity which is also present within the supply chain management hierarchy described previously in Section 2. Before jobs are scheduled and released to the shop floor, the arrival of jobs to the operational management level within the factory is simulated. Typically, this task is performed on the tactical level, which is modeled in the simulation as the *Job generator* component. Once jobs are generated, a schedule can be created which is generally a task performed on the operational management level. Within this simulation system, this process is contained within the *Scheduler* constituent. Afterwards, the jobs are sent to the shop floor where they are produced by the assembly lines, i.e., the manufacturing execution level. This highly stochastic process is modeled in the *Shop floor* component. At last, the *Rescheduler* monitors the progress at the shop floor through key performance indicators and may trigger the *Scheduler* when deemed necessary, e.g. in case significant delays from the original schedule are observed. This way, the *Rescheduler* establishes a closed loop between the operational management level and the manufacturing execution level. A schematic overview of the entire system and its elements is shown in Figure 1. Next, the content of each component is explained in further detail.

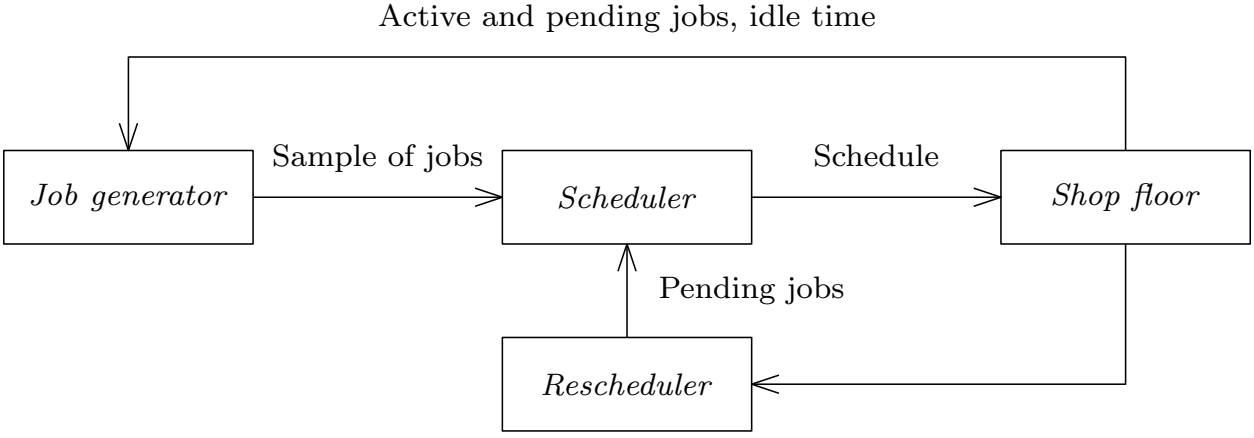

Figure 1: Schematic overview of the complete simulation model.

Job generator At regular intervals of N days, this module creates a batch of jobs which is sent to the *Scheduler* component. This way, it mimics the interaction of the tactical with the operational management, which usually occurs on a weekly or biweekly basis (i.e., $N = 7$ or $N = 14$). As mentioned, the tactical management layer uses rough-cut models to estimate the available capacity. Here, a rough estimate is made

in three steps: (i) determine the maximum capacity, (ii) estimate the remaining work from the previous period and (iii) obtain the idle time during the previous period. Note that the capacity is measured in time. The maximum capacity represents the capacity if the system is empty, i.e., the number of machines multiplied by the length of the N day period. However, when the estimate is made, there may be unfinished jobs in the system, and some capacity must be allocated to these unfinished jobs in the coming period. To account for this, the expected duration of these jobs is subtracted from the maximum capacity. Similarly, some machines may already be idle for some time, meaning that the previous batch of jobs was not sufficient to fill the available capacity for N days. Thus, the workload of the next sample should be increased. Hence, for every machine, this idle time (if any) is added, yielding a final estimate for the total capacity to fill: the target capacity C.

Now that the target capacity is known, the next step is to fill this capacity with jobs. This is done by randomly sampling jobs from a sample pool, which consists of historical jobs. Given the large number of historical realizations, this pool is a realistic representation of the size distribution as well as the type mix, referring to different product types. Sampling a job from this pool is done is two steps. First, the product type p is sampled. Then, the quantity Q is sampled from a subset of the pool that only contains historical realization of jobs for product type p. This is to account for the fact that some product types always come as small jobs, whereas others (so-called high-runners) always come as large jobs. Now, given p and Q, its effective process time (EPT) can be estimated. The effective process time, or simply the total duration of the job from start to finish, constitutes three components: (i) setup time, (ii) process time and (iii) down time:

$$EPT = t_{setup} + t_{process} + t_{down}$$

where the setup time t_{setup} is sequence dependent, i.e., it depends on the job itself as well as on the job previously processed on the machine. The process time $t_{process}$ and the downtime t_{down} depend on the machine and the product type. However, as the schedule is yet not constructed, the machine assignment is unknown at this point. Therefore, an estimate for the EPT (denoted as $EPT^{est.}$) is calculated as follows:

$$EPT^{est.} = \frac{t_{setup}^{min.} + Q/\bar{v}_p}{1 - \bar{f}_{down,p}} \tag{1}$$

where $t_{setup}^{min.}$ is the minimal setup time, i.e., the time that is always incurred regardless of the product type or scheduling decisions, $\bar{v}_p$ is the average process speed of jobs for product type p over all machines measured in products per time unit, and $\bar{f}_{down,p}$ is the average fraction of the time that a job for product type p was in the down state over all machines. Although almost certainly $EPT^{est.}$ will differ from the eventual simulated realization, a rough estimate is sufficient at this stage and also mimics the actual way of working at the tactical management level. The estimate of the sampled job is subtracted from the total capacity that needs to be filled and the sampling procedure continues until the entire sample of jobs fills this total capacity. The complete procedure in summarized in Algorithm 1. When the sample is complete, it is sent to the *Scheduler*.

Scheduler After the sample of jobs is received from the *Job generator*, a new schedule is created. This schedule includes the new jobs together with the unfinished jobs that were already scheduled previously. As jobs are non-preemptive, only jobs that are unfinished and not started are included. The remaining time of a job that is currently in process is accounted for by the scheduler. The scheduling problem is characterized by unrelated parallel machines, sequence and machine dependent setup times, machine eligibility constraints and job specific release and due dates. The problem is treated as deterministic, meaning that the scheduler does not consider the stochasticity of the shop floor in making scheduling decisions. However, contrary to the rough capacity estimate made earlier by the higher management, the scheduler does consider machine dependencies. For this purpose, the scheduler uses a machine learning (ML) model that accurately predicts the EPT of a job, given its characteristics as well as the position in the schedule (machine, predecessor, successor). This ML model was developed in a previous study, where

Algorithm 1: Job sampling procedure

Input: Target capacity C
Result: Sample of jobs
while $C > 0$ **do**
  $p \leftarrow$ Uniform random sample from historical data
  $Q \leftarrow$ Uniform random sample from historical data of product type p
  Add job for product type p and quantity Q to sample
  $EPT^{est.} \leftarrow$ Equation (1)
  $C \leftarrow C - EPT^{est.}$
end

it was exported to the open neural network exchange (ONNX) format, facilitating facile inclusion of the ML model into the scheduler (Shridhar et al. 2020). Furthermore, the scheduler itself utilizes a hybrid genetic algorithm (HGA) that was also developed in a previous study and specifically designed for the problem at hand (Adan et al. 2018). A genetic algorithm is a metaheuristic optimization method that simulates the process of natural evolution. In this algorithm, a population of candidate solutions within the search space, so-called individuals, evolves toward better solutions through an iterative evolutionary process. Each candidate solution in the population represents a schedule. Eventually, the best schedule from the population is chosen. In this case, the objective is to minimize the makespan. Hence, the schedule with the lowest makespan value is selected. This schedule is sent to the shop floor.

Shop floor The shop floor receives a schedule from the operational management level and follows this during execution. Specifically, the shop floor is modeled as a discrete event simulation that consists of multiple parallel machines with queues (one queue for each machine). The jobs in each queue and their sequence are dictated by the received schedule. When a machine is available and the queue is not empty, the next job is started. At this point, the "real" EPT (denoted as EPT^{real}) is determined:

$$EPT^{real} = \frac{t_{setup} + Q/v_p}{1 - f_{down,p}} \tag{2}$$

which is very similar to Equation (1) except that here t_{setup} is the actual setup time, and v_p as well as $f_{down,p}$ are random samples from the respective empirical distributions (whereas Equation (1) uses the averages of these distributions). Note that the setup time is deterministic in Equation (2). Although in reality the setup time is a stochastic value, attempts to extract this value from the available historical data were not successful. Hence, the sequence- and machine-dependent deterministic setup times used by the operational management are also used here.

Rescheduler While the schedule is executed at the shop floor, progress is continuously monitored. Specific key performance indicators (KPIs) can quantify progress and deviation from the original schedule. These can in turn be used by heuristics or other methods that decide when to trigger a new scheduling action in order to optimize a certain objective, e.g. throughput or tardiness. When a decision is taken to perform a rescheduling action during schedule execution, the *Scheduler* is triggered again. A new schedule will then be generated based on the jobs that at that time are still waiting in the queue (recall that jobs are non-preemptive). From that point onward, the *Shop floor* will adhere to the newly created schedule until a new rescheduling decision is made or the end of the N day period is reached (which means that a new sample of jobs is released from the *Job generator* that also triggers the *Scheduler*).

3.3 Simulation Settings

Several settings have to be specified before a simulation can be executed. First, regarding the *Job generator*, for all simulations in this work the length of the interval between subsequent job releases is set to a week, as is common in practice.

The second aspect of the settings concerns the *Scheduler* module. As explained in the previous section, this module uses an HGA. The number of evolutionary cycles that is necessary for the objective to converge depends on the size of the problem instance. Based on preliminary experiments, the number of iterations is set to 400, as this appeared sufficient to always reach convergence. Furthermore, the algorithm is always run four times (in parallel). In the end, the best schedule in terms of the objective is selected.

Third, the type of the simulation needs to be specified. To obtain statistically accurate results, either the length of the simulation must be sufficiently long or, if this is not the case, a large number of replications is required (Banks 1998; Law et al. 2007). The choice is made to perform the former as it is of interest to know the long-run average of the performance measures. As the simulation starts off completely empty, the behavior during the first few weeks is slightly different compared to the steady-state behavior. To safely eliminate this start-up effect, every simulation starts with a warm-up period of 20 weeks (meaning that during this period all the observations are discarded). After the warm-up phase, the simulation continues for another 400 weeks. The relative confidence interval (RCI) is defined as the confidence interval normalized with the mean. For the relevant performance measures (which will be introduced shortly), preliminary experiments have shown that with these settings the RCI is sufficiently low, i.e., well below 0.05.

3.4 Validation

The purpose of the simulation model described in the previous section is to mimic the real-world scheduling and production, in combination with the interactions between the various management layers. This section evaluates whether the simulation model fulfills its purpose. First, the behavior of the job generator is evaluated, and second, the shop floor is addressed.

Workload When the *Job generator* releases a sample of jobs (a workload), its intention is to occupy all the available capacity at the *Shop floor* from that moment up to the next time it generates a sample of jobs, which are separated by a specifiable regular time interval. This interval is set to a week (which is common in practice) and a simulation is run. During this simulation, two quantities are tracked: (i) the workload released by the *Job generator*, and (ii) the workload at the *Shop floor* (i.e., active jobs plus jobs in the queue). In both cases the workload is estimated using $EPT^{est\cdot}$ from Equation (1). Figure 2 shows a snapshot of both quantities at an arbitrary time during the simulation. The total workload in the system (blue) increases instantaneously at regular intervals, corresponding to the moment where the *Job generator* releases new jobs. Then, the workload at the *Shop floor* decreases steadily over the course of a week as jobs are processed. However, at the end of each weekly period, right before the new jobs are released, there is some workload remaining. It can clearly be seen that the *Job generator* adjusts the next release of jobs (red) accordingly, as is the intended behavior.

Additionally, Figure 3 shows the number of active machines over the time period. Clearly, some machines become idle near the end of a week, despite the fact that the workload on the shop floor is far from zero (recall Figure 2). This suggests that during the course of the week, the stochasticity of the system developed an imbalance in the distribution of the workload over the machines, which is a promising prospect for the potential of rescheduling.

Effective processing speed The effective process time at the shop floor is modeled by Equation (2). To visualize and verify whether this model is in accordance with the historical realizations, these realizations need to be compared to the simulated values. To enable this, all realized EPT values are first normalized by the product quantity Q, yielding the effective process *speed* (EPS). This simulation produced a total of 66 000 values while the historical data contains 20 000 observations. Figure 4 shows box-and-whisker plots of the historical and the simulated values for two different machine types. In addition, the historical and

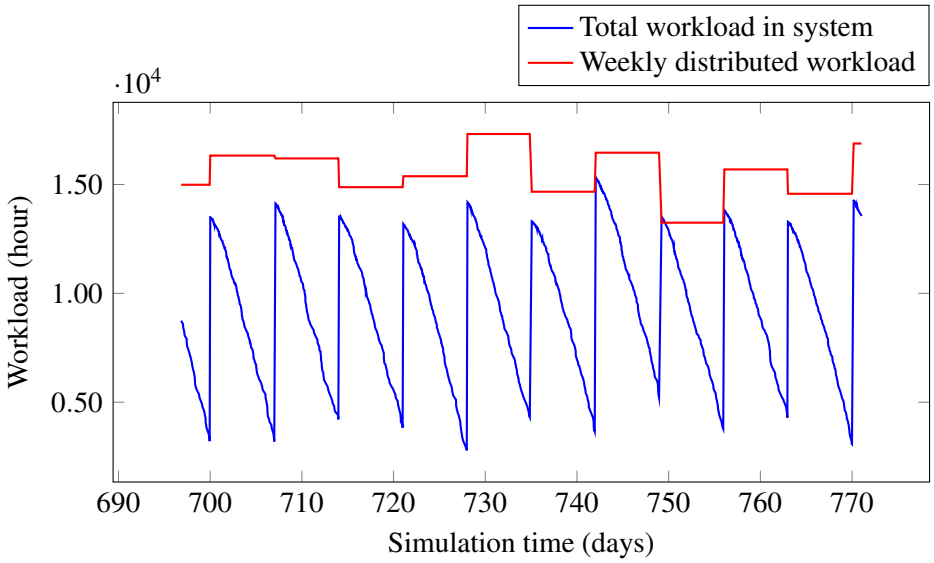

Figure 2: A snapshot of the released workload according to the *Job generator* and the workload perceived by the *Shop floor*. In this example, the *Job generator* is triggered every week.

simulation data for each machine type is sorted based on a product characteristic that has a high influence on the speed. Clearly, the shape of the distributions and the medians are very similar for each machine type and product characteristic. In most cases, the variation of the simulated values is slightly higher compared to the history. This is not entirely surprising, as it may be due to the fact that a simulation requires two values (the process speed v_p and the average fraction of the time in the down state $t_{down,p}$) to be sampled, which may increase variation.

4 COMPUTATIONAL EXPERIMENTS

This section focuses mainly on the rescheduling module, previously introduced in Section 3.2. In its most abstract sense, this module continuously monitors the shop floor progress and triggers rescheduling when deemed necessary. Progress can be described by various performance indicators, and rescheduling does not necessarily affect all the jobs, but may be limited to a selected subset. In other words, various heuristics can be implemented in this module. To demonstrate this setup and determine the value of rescheduling, a series of simple time-based rescheduling heuristics are implemented. Then, to compare these heuristics mutually, a simulation is run for each heuristic. Also, a simulation is run without any rescheduling to serve as a general benchmark.

The job generator evaluates the remaining workload in the system at regular time intervals and samples new jobs such that the system contains a specified target workload, as explained earlier. During the interval, the rescheduling module is kept active. In case of a time-based heuristic, rescheduling is triggered at fixed times, regardless of the progress at the shop floor. These triggers are spread over time, such that the interval contains H equidistant triggers, as illustrated in Figure 5. Varying H leads to a series of time-based heuristics.

To evaluate the potential of these rescheduling heuristics, multiple simulations are conducted. For each heuristic, the settings stated in Section 3.3 are used. Throughout the course of 400 weeks, every week, several measures are recorded, namely: the total quantity that was processed, the total idle time, the average tardiness and the total number of rescheduled jobs. All these measures are of significant practical relevance. The results for each heuristic, characterized by the number of triggers H within a weekly period, are summarized in Figure 6. Here, the case where H equals zero refers to the scenario without rescheduling. First of all, it can be seen that rescheduling more frequently (from left to right) increases the number of

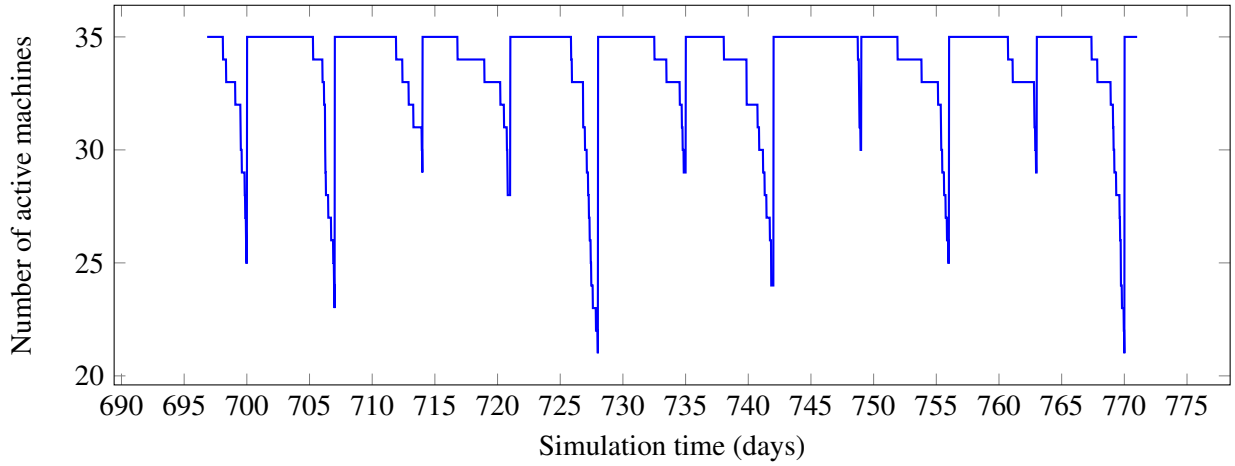

Figure 3: A snapshot of the number of active machines on the *Shop floor*. In this example, the *Job generator* is triggered every week, i.e., $H = 7$.

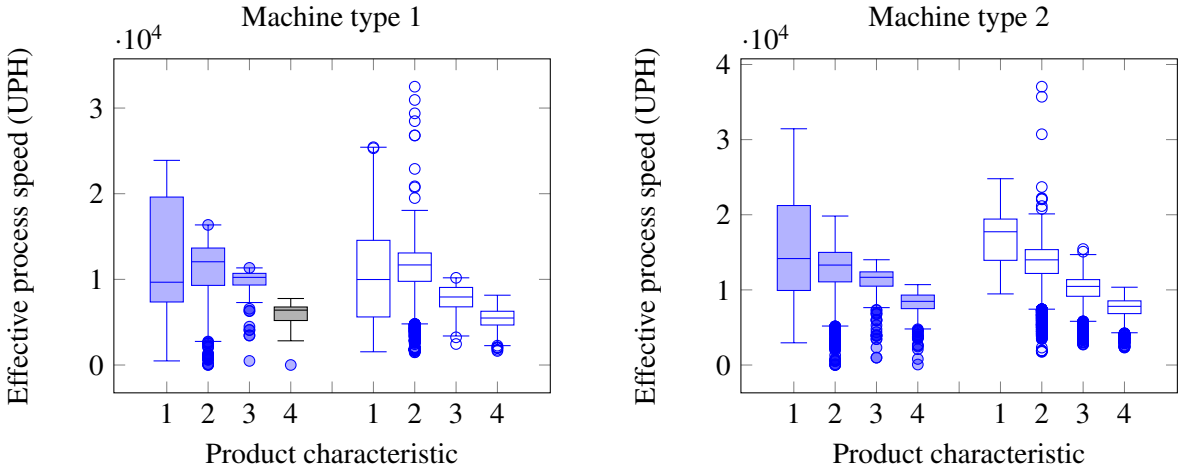

Figure 4: Distribution of the effective process speed from historical data (filled) and simulation data (non-filled) for two different machine types.

products that are produced (Figure 6a), which is perfectly in line with the decreasing trend of the idle time (Figure 6b). More precisely, the throughput difference between the scenario without rescheduling and the case where rescheduling is triggered every day is 2.15%. While this appears as a small gain, in the semiconductor industry this leads to large revenue benefits. The tardiness also improves significantly, see Figure 6c. A large difference is observed between the scenario without rescheduling ($H = 0$) and all cases with rescheduling ($H > 0$). Interestingly, as rescheduling is triggered more frequently, the tardiness increases slightly. A probable explanation for this phenomenon is that as rescheduling reduces the idle time, it fills this idle time with jobs. Although this is beneficial for the throughput, this also poses a risk, as a job may unexpectedly take longer than predicted. As a result, the job may incur tardiness itself, but also translates the delay to jobs later in the queue. At last, Figure 6d shows the number of jobs involved in rescheduling. To place these numbers in perspective: the total number of jobs that comprise a weekly workload lies around 160 to 170. Frequent rescheduling may not be desired by the shop floor workers, since it may disrupt an orderly way of working and make preparations useless – effects that are not included in

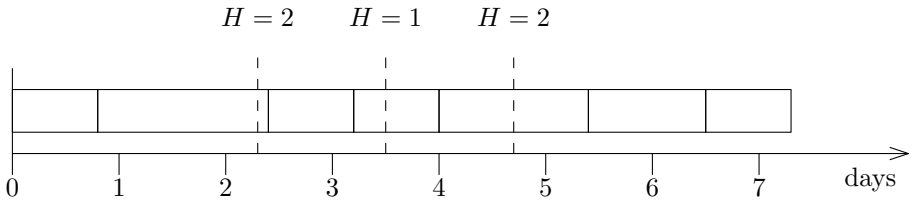

Figure 5: Time based heuristics with one (H is 1) and two (H is 2) triggers in case the job generator is triggered every week. A single machine is considered and the blocks represent jobs.

this model. All in all, these experiments clearly quantify the potential benefit that can be achieved through rescheduling.

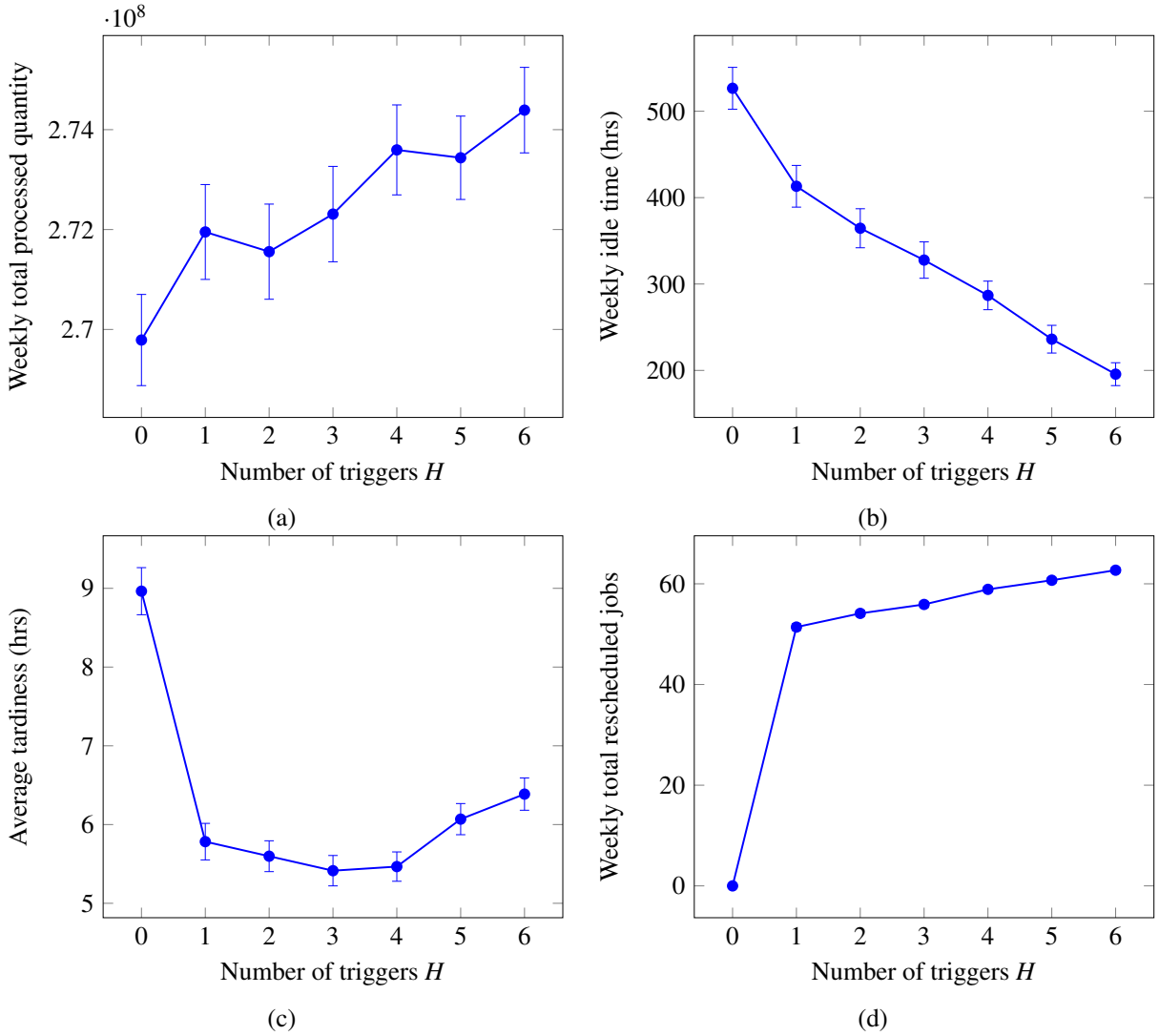

Figure 6: Results for the various time based heuristics each characterized by a different number of rescheduling triggers H: (a) weekly total processed quantity, (b) the weekly total idle time, (c) the average tardiness and (d) the weekly total number of rescheduled jobs. The bars indicate the 95% confidence interval.

5 CONCLUSIONS AND FUTURE WORK

We consider a back-end semiconductor manufacturing setting with a focus on production scheduling. Motivated by the real-life observation that actual production schedules executed on the shop floor often deviate from the planned schedules, we propose an approach that can easily be followed in practice to minimize the effect of such deviations by effectively triggering the construction of new production schedules, referred to as rescheduling in the paper. The approach uses a simulation model that is validated with real-life data from a semiconductor manufacturer. An important feature of the simulation model is its ability to mimic not only the shop floor but also the interactions with higher management levels. In this way, the amount of jobs released to the shop floor can be controlled for the purpose of a more effective rescheduling. In our study, we use the simulation model to quantify the benefit from a time-based rescheduling heuristics.

Despite the improved performance measures, the rescheduling heuristics proposed in this study considers the entire schedule being executed on the shop floor, which is not desirable in practice. A future research direction is to develop improved rescheduling heuristics that can identify machines that need rescheduling (e.g. delayed or break down) and then reschedule only a specific part of the schedule to minimize the disturbance to the shop floor.

REFERENCES

Adan, J., A. Akcay, J. Stokkermans, and R. V. d. Dobbelsteen. 2018. "A Hybrid Genetic Algorithm for Parallel Machine Scheduling at Semiconductor Back-end Production". *Association for the Advancement of Artificial Intelligence*.

Adan, J. and Deenen, P.C. and Geurtsen, M. 2020. "C# Simulation Library". https://github.com/JelleAdan/CSSL. Accessed: January 2021.

M.P. Albers 2022. "The Autonomous Batch Scheduler". https://research.tue.nl/en/studentTheses. Accessed: April 2022.

Banks, J. 1998. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons.

Chen, C., M. Fathi, M. Khakifirooz, and K. Wu. 2022. "Hybrid Tabu Search Algorithm for Unrelated Parallel Machine Scheduling in Semiconductor Fabs with Setup Times, Job Release, and Expired Times". *Computers and Industrial Engineering*:107915.

Deenen, P. C., J. Adan, and A. Akcay. 2020. "Optimizing Class-Constrained Wafer-to-Order Allocation in Semiconductor Back-End Production". *Journal of Manufacturing Systems* 57:72–81.

Fortune 2020. "Semiconductor Market Size, Share & Covid-19 Impact Analysis". https://www.fortunebusinessinsights.com/semiconductor-market-102365. Accessed: April 20th 2022.

Geurtsen, M., J. Adan, and A. Akcay. 2022. "Integrated Maintenance and Production Scheduling for Unrelated Parallel Machines with Setup Times". School of Industrial Engineering, Eindhoven University of Technology, Working paper.

Law, A. M., W. D. Kelton, and W. D. Kelton. 2007. *Simulation modeling and analysis*, Volume 3. McGraw-Hill New York.

Liu, A., J. Fowler, and M. Pfund. 2016. "Dynamic Coordinated Scheduling in the Supply Chain Considering Flexible Routes". *International Journal of Production Research* 54(1):322–335.

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.

Shridhar, A., P. Tomson, and M. Innes. 2020. "Interoperating Deep Learning Models with ONNX.jl". In *Proceedings of the JuliaCon Conferences*, edited by R. Anantharaman, C. Bauer, M. Besançon, V. Churavy, and A. Smith, Volume 1, 59. Open Journals.

Steven, M. 2004. "Hierarchical Planning Structures in Supply Chain Management". In *Modern Concepts of the Theory of the Firm*, edited by G. Fandel, U. Backes-Gellner, M. Schlüter, and J. E. Staufenbiel, 301–312. Berlin: Springer.

AUTHOR BIOGRAPHIES

JELLE ADAN is a doctoral candidate in the School of Industrial Engineering at the Eindhoven University of Technology and a co-founder of Atlas4 Technologies. His current research interests are supply chain, manufacturing and chemical process optimization, and data science. His email address is jelle.adan@protonmail.com.

MICHAEL GEURTSEN is a doctoral candidate in the School of Industrial Engineering of the Eindhoven University of Technology and a Sr. Business Process Analyst at Nexperia. His current research interests are in the area of modeling and optimization of maintenance in manufacturing systems. His email address is michaelgeurtsen@protonmail.com.

ALP AKCAY is an Associate Professor in the School of Industrial Engineering at Eindhoven University of Technology. He received his Ph.D. in Operations Management and Manufacturing from Carnegie Mellon University. His current research in-

terest is data-driven decision making with applications in manufacturing and maintenance. His email address is a.e.akcay@tue.nl.

MARC ALBERS was a student in the School of Mechanical Engineering at Eindhoven University of Technology. He recently obtained his master's degree in Mechincal Enigneering and Manufacturing System Engineering and started as a Manufacturing Engineer at Philips Healthcare. His current research interest are manufacturing process optimization and simulation. His email address is marc.albers@hotmail.com.

JOHN FOWLER is the Motorola Professor of International Business in the Supply Chain Management department in the W.P. Carey School of Business at Arizona State University. His research interests include discrete event simulation, deterministic scheduling, multi-criteria decision making, and applied operations research with applications in semiconductor manufacturing and healthcare. He has published over 135 journal articles and over 100 conference papers. He was the Program Chair for the 2008 Winter Simulation Conference and currently serves on the WSC Board of Directors. His email address is john.fowler@asu.edu.