Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

SCHEDULING JOBS IN A TWO-STAGE HYBRID FLOW SHOP WITH A SIMULATION-BASED GENETIC ALGORITHM AND STANDARD DISPATCHING RULES

Benjamin Rolf
Tobias Reggelin
Abdulrahman Nahhas
Marcel Müller

Otto von Guericke University Magdeburg
Universitätsplatz 2
Magdeburg, 39106, GERMANY

Sebastian Lang

Fraunhofer Institute for
Factory Operation and Automation (IFF)
Sandtorstraße 22
Magdeburg, 39106, GERMANY

ABSTRACT

The paper proposes a simulation-based hyperheuristics approach to generate schedules for a two-stage hybrid flow shop scheduling problem with sequence-dependent setup times. The scheduling problem is derived from a company that is assembling printed circuit boards. A genetic algorithm determines sequences of standard dispatching rules that are evaluated by a discrete-event simulation model minimizing a multi-criteria objective composed of makespan and total tardiness. To reduce the computation time of the algorithm a dispatching rule-based chromosome representation is used containing a sequence of dispatching rules and time intervals in which the rules are applied. Different experiment configurations and their impact on solution quality and computation time are analyzed. The optimization model generates efficient schedules for multiple real-world data sets.

1 INTRODUCTION AND LITERATURE REVIEW

A hybrid flow shop or flexible flow shop is a well-known scheduling problem that requires the sequencing of n jobs onto m machines with the objective of minimizing a certain objective function. The hybrid flow shop consists of c production stages in series with identical parallel machines on each stage. Each job j has to be processed on one of the parallel machines on every stage in the same order (Pinedo 2016; Baker and Trietsch 2009). Hybrid flow shop scheduling problems belong to the class of NP-hard optimization problems (Garey and Johnson 1996; Gupta 1988). Hence, no polynomial time algorithm exists to solve the problem and metaheuristics are often an appropriate tool to find feasible solutions (Talbi 2009). Genetic algorithms, introduced by Holland (1975) and Goldberg (1989), are common metaheuristics that have been applied for solving scheduling problems frequently. They transfer principles of the evolutionary theory to mathematical optimization problems.

Recently, the new class of hyperheuristics evolved which intelligently administrates and assigns low-level construction heuristics to search the solution space of combinatorial optimization problems (Burke et al. 2013). The most frequently used simple construction heuristics in production scheduling are dispatching rules due to their ease of implementation and fast computation time. Dispatching rules make a scheduling decision whenever a machine becomes idle by computing a priority value and picking the job with the minimal value from the corresponding queue (Blackstone et al. 1982; Haupt 1989). Therefore, many hyperheuristic approaches in production scheduling are based on dispatching rules that are administrated by high level metaheuristics like genetic algorithms. Genetic algorithms for assigning dispatching rules need specific chromosome representations to deal with scheduling problems (Talbi 2009). Cheng et al. (1996) list

multiple possible representations for genetic algorithms and separate them in direct and indirect approaches. The permutation representation or job-based representation is the most common direct representation for scheduling problems which depicts the order of jobs for every machine. Each job can only appear exactly once in the permutation representation (Talbi 2009; Cheng et al. 1996). The dispatching rule-based representation was first used by Dorndorf and Pesch (1995) and forms the basis for the implementation of hyperheuristics. In the dispatching rule-based representation, solutions are represented as sequences of dispatching rules that are called one after another by the simulation model to transform them into a job schedule (Vázquez-Rodríguez and Petrovic 2010; Glass and Potts 1998; Cheng et al. 1996). It has been successfully applied to various scheduling problems.

Huang and Süer (2015) developed an hybrid simulation-based genetic algorithm with dispatching rules and fuzzy satisfaction levels and used it to generate schedules for various artificial data sets with 5-10 machines and 10-20 jobs in a job shop. The chromosomes of the genetic algorithm are composed of individual sequences of dispatching rules for each machine. A discrete-event simulation model evaluates the sequences of dispatching rules and switches them in predefined and fixed time intervals. When the sequences finish, they start all over again. Additionally, the authors used fuzzy satisfaction levels to determine the conformity of a schedule to four different objectives.

Vázquez-Rodríguez and Petrovic (2010) proposed a genetic algorithm using an advanced version of the dispatching rule-based representation to generate schedules in a job shop scheduling problem. They additionally introduced an integer value for each dispatching rule in the sequence that indicates the number of jobs that the machine has to process using the called dispatching rule. This representation is able to dynamically assign dispatching rules and find good schedules for various multi-objective scheduling problems. They also compared the solutions to the permutation representation and concluded that their approach obtains higher quality solutions. Ochoa et al. (2009) analyze four artificial instances of a hybrid flow shop with 50-100 jobs and 5-20 stages with a similar genetic algorithm as Vázquez-Rodríguez and Petrovic (2010). They also concluded that the dispatching rule-based representation outperforms the permutation representation.

Zhang et al. (2013) developed a hyperheuristic genetic algorithm focusing on minimizing due date related performance measures and generating non-delay schedules. Their genetic algorithm searches for good sequences of dispatching rules in a job shop environment involving eight different dispatching rules and a discrete-event simulation model. They used the algorithm on several benchmark data sets and compared the results to other algorithms.

This paper is structured as follows. Section 2 deals with the definition of the scheduling problem including the machine environment, the job characteristics and the objective function for the genetic algorithm. The following section presents our simulation-based genetic algorithm with the implemented dispatching rule-based representation. Furthermore, the parameters of the simulation model for the experiment and genetic algorithm configuration and the output parameters are shown. In section 4 we conduct experiments by varying certain parameters of the simulation model and present exemplary solutions of the scheduling problem. Finally we conclude our findings in section 5 and shortly discuss future research directions.

2 PROBLEM DEFINITION

This paper solves the hybrid flow shop scheduling problem at the example of an electronics manufacturing services company that produces printed circuit boards (PCB). The company provided us with four data sets with $n = 150$-200 jobs. The setup times are sequence-dependent. The following sections describe the machine environment, job characteristics and objectives of the scheduling problem.

2.1 Machine Environment

The hybrid flow shop production system (Figure 1) consists of two production stages with four identical parallel machines on the first stage and five identical parallel machines on the second stage ($m = 9$). The

surface mounting devices (SMD) on the first stage perform the main assembly process using surface mounting technology (SMT). The machine first places the PCBs, applies soldering paste and finally fabricates the solder connection.

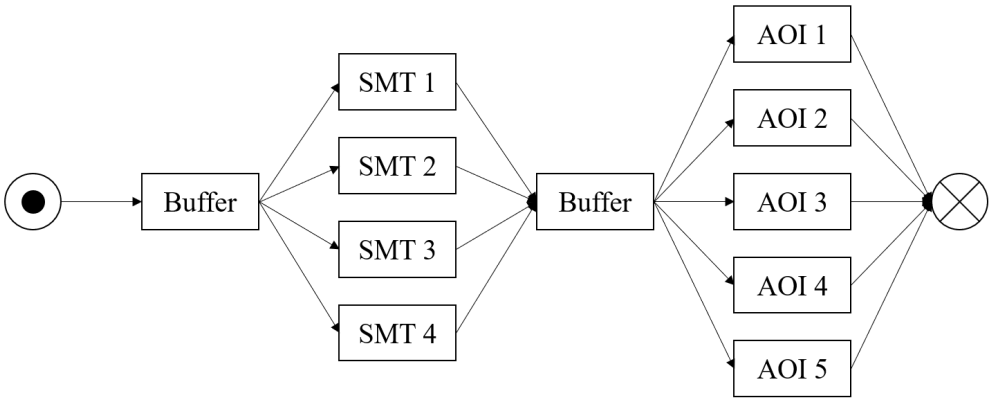

Figure 1: Two-stage hybrid flow shop of an electronics manufacturing services company

The suppliers of the company deliver the raw materials for the SMT process on material rolls which must be placed on setup carts before processing on the first production stage. Due to the enormous number of about 700 PCB variants and the limited number of setup carts the material roll and the tools on the setup carts have to be changed frequently. To reduce the number of possible setup cart configurations the company clustered similar variants into about 50 product families. Each setup cart is always prepared for the manufacturing of all jobs of a product family. Switching the setup cart configuration is considered as the setup process of the SMT production stage. This process depends on the sequence of jobs respectively the family that was previously processed. Therefore we have to consider sequence-dependent setup times with two possible scenarios on the SMT production stage:

1. Job j has the same product family f_j as the previously processed job k ($f_j = f_k$). The configuration of the setup cart remains the same. We define this situation as a **minor setup**. For a minor setup the worker provides the PCBs of the new job to the machine, changes the soldering paste template and loads the machine program. The minor setup process takes $s_{j,k} = 20$ minutes in total.
2. Job j has another product family f_j as the previously processed job k ($f_j \neq f_k$). The worker has to change the configuration of the setup cart. We define this situation as a **major setup**. For a major setup the worker additionally changes the setup cart configuration which takes additional 45 minutes. Therefore the major setup process takes $s_{j,k} = 65$ minutes in total.

The five machines of the second production stage perform an automatic optical inspection (AOI) to check the quality of each PCB. The machine setup on the AOI stage has to be done before each job and includes the loading of the testing program and the validation of each PCB. The setup process takes $u_{j,k} = 25$ minutes and is sequence-independent.

2.2 Job Characteristics

The following attributes characterize each job:

1. The processing times p_{cj} for job j on production stage c with $c \in \{SMT, AOI\}$.
2. The due date d_j for job j.
3. The product family f_j for job j.

All jobs with their related due date, product family and processing times are previously known for the next production period. However, our approach also works when jobs arrive continuously. Every time a new job arrives, the new job is added to the production program and a new optimization run starts taking into account the next possible point of time (e.g. next shift, next day), when the production schedule can be actually changed. Furthermore, the simulation model which evaluates the schedule needs to be initialized with the current production status. To represent the actual situation in the company we have to determine some additional problem-specific restrictions. It is not allowed to mount the same product family on two or more different machines at the same time because the company cannot setup two setup carts with the same configuration simultaneously due to the limited equipment. Additionally, we assume that every machine is setup for the family of the first job at time zero to match the assumptions of other authors that solved this scheduling problem previously (Rolf et al. 2020; Aurich et al. 2017; Nahhas et al. 2016). The general assumptions for static and deterministic scheduling problems of Rolf et al. (2020) also apply to this paper.

2.3 Objectives

The PCB company tries to optimize two goals with its scheduling policy which are a preferably minimal makespan and the adherence of all due dates. Currently the schedulers of the company try to reach this goals with a pure family setup policy which aims to keep the major setups of the SMT stage at an absolute minimum. This policy occasionally leads to a violation of job due dates.

We want to reconsider the scheduling policy to a more flexible approach that is not necessarily based on a strict major setup avoidance, so the number of major setups is not included in the objective function. Furthermore, the genetic algorithm should use the objective function as its fitness function. Hence, it needs to be a numerical objective that states the quality of a schedule and can be quickly computed by a simulation model.

Well-known numerical objectives are the makespan and the total tardiness. The makespan C_{max} is widely used for scheduling problems and is defined as the maximum completion time C_j of all jobs

$$C_{max} = max\{C_j | j = 1, ..., n\}.$$

The total tardiness T is also a commonly used objective which minimizes the tardiness of jobs in a schedule. It is defined as the sum of tardiness of all jobs with the tardiness being the difference between completion time C_j and due date d_j of job j

$$T = \sum_{j=1}^{n} max\{0, C_j - d_j\}.$$

In our case, the main objective of the PCB company was to reduce the tardiness as far as possible and at the same time to have a low makespan. Our Experiments have shown that using the sum of makespan and total tardiness as the fitness function for the genetic algorithm achieved the best results. Additionally, the permitted tardiness T_{per} determines the maximum tardiness a schedule can contain without being penalized in the objective function. The fitness F of an individual I, or more specifically, a schedule is defined as

$$F(I) = C_{max} + max\{0, T - T_{per}\}.$$

3 SIMULATION-BASED GENETIC ALGORITHM

The optimization model proposed in this paper comprises a genetic algorithm, a discrete-event simulation model and multiple standard dispatching rules. The genetic algorithm serves as a hyperheuristic that is hierarchically superordinate to the simulation model and controls the evolution process. The simulation model takes the evaluation function in the optimization process and evaluates the candidate solutions specified by the genetic algorithm (Gosavi 2015; Banks et al. 2010). The standard dispatching rules are used as low level construction heuristics for an easy and efficient representation of feasible solutions. The

entire solution approach is based on the idea that each standard dispatching rule is superior to others in certain situations. If the hyperheuristic switches the dispatching rules at the right points in time even complex multicriteria objectives can be optimized.

Contrary to similar papers of Huang and Süer (2015), Zhang et al. (2013), Vázquez-Rodríguez and Petrovic (2010) and Ochoa et al. (2009) we address a real-world machine environment with sequence-dependent setup times that needs a specific set of dispatching rules. We implemented both parts of the optimization model with the discrete-event simulation software Tecnomatix Plant Simulation using the standard material and information flow libraries and the genetic algorithm library.

3.1 Dispatching Rule-based Representation

Scheduling problems in genetic algorithms may be encoded in different representations such as the permutation representation or the dispatching rule-based representation. The permutation representation comes with the disadvantage of a lot of genes resulting in a long computation time for datasets with a lot of jobs (Vázquez-Rodríguez and Petrovic 2010; Ochoa et al. 2009). Therefore, the permutation representation is not feasible to tackle the complexity and size of real-world datasets with a few hundred jobs in terms of computation time. Instead, we implemented an advanced version of the dispatching rule-based representation that consists of a sequence of dispatching rules to reduce the number of genes respectively the computational complexity. The genetic algorithm hyperheuristic chooses dispatching rules out of a predefined set of rules. The algorithm determines a dispatching rule for the SMT stage and a dispatching rule for the AOI stage and changes the rules flexibly during the time of job processing. Figure 5 shows an example of the flexible switching points and the sequence of dispatching rules. The sequence of dispatching rules defines the sequence of jobs indirectly by computing priority values for each job in the buffer and picking the job with the minimal value. We used the following eight dispatching rules, subdivided into job-based and family-based dispatching rules:

- **EDD**: Earliest Due Date (job-based)
- **MST**: Minimum Slack Time (job-based)
- **CR**: Minimum Critical Ratio (job-based)
- **SPT**: Shortest Processing Time (job-based)
- **SST & EDD**: Shortest Setup Time & Earliest Due Date (family-based)
- **SST & MST**: Shortest Setup Time & Minimum Slack Time (family-based)
- **SST & CR**: Shortest Setup Time & Minimum Critical Ratio (family-based)
- **SST & SPT**: Shortest Setup Time & Shortest Processing Time (family-based)

Rolf et al. (2020) defined all used dispatching rules mathematically in their previous work. The job-based rules are four commonly used standard dispatching rules without any modifications while the four family-based rules are specifically designed to deal with the sequence-dependent setup times of the scheduling problem.

To create the family-based rules, we use the dispatching rule shortest setup time (SST) and combine it with the four job-based rules serving as tiebreakers. At first, the SST rule separates all jobs in the buffer by their product families into two groups: Jobs that need a minor setup and jobs that need a major setup. Only when multiple jobs have the same setup time or more specifically the same product family the tiebreak rules subdivide the jobs to make distinct scheduling decisions. Generally, the family-based dispatching rules prefer jobs of the mounted family over jobs of different families to minimize the number of major setups and to reduce the total setup time.

The set of dispatching rules for the SMT stage contains job-based and family-based dispatching rules while the genes associated with the AOI stage can only take the job-based rules because sequence-independent setup times cannot be minimized. The setup times on the AOI stage always remain the same.

Figure 2 shows an example of the rule-based representation of the genetic algorithm. Every set refers to one time interval and needs three genes to define the interval distinctly. The switching point determines the point in time where the dispatching rules are switched and the dispatching rule combination contains the dispatching rules for both stages which are introduced at that point in time. The first set exceptionally consists of only two genes because it defines the initial dispatching rule combination that always has to be introduced at time zero. Therefore, the first set is called "Set 0" and the switching point is not in the genes of the genetic algorithm.

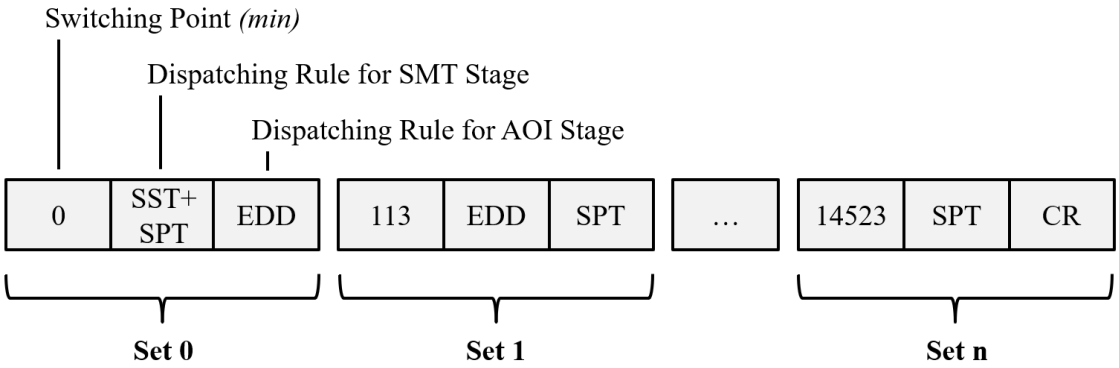

Figure 2: Dispatching rule-based representation of the genetic algorithm.

3.2 Model Configuration and Output

The configuration parameters are split into two groups. The experiment configuration contains all parameters that define the specific scheduling problem in detail and form the basis for the optimization. The genetic algorithm configuration uses the given parameters of the Plant Simulation library to define the general optimization procedure. The following list shortly describes all input parameters that can be used to adjust the experiment configuration:

1. The *permitted tardiness T_{per} (min)* determines the maximum tardiness that the schedule can contain without triggering a penalty in the objective function.
2. The *switching period S_{period} (min)* is the maximum time period in which a rule can be switched. If the approximate makespan of the data set is known this parameter can be used to decrease the solution space and speed up the optimization run.
3. The length of the *planning period t (days)* defines the number of days that the genetic algorithm should find a schedule for. The input data set has to be provided in a separate table. If the data set contains a lot of jobs this parameter can filter the jobs to schedule only the first few of them.
4. The *number of switching points n (number)* defines the number of sets that is used in the representation of the genetic algorithm, see Figure 2. A high number of switching points leads to more genes and longer computation time.
5. The *saving points G_{save} (table)* define the generations of the genetic algorithm in which the currently best solution is saved in the result table.

The configuration of the genetic algorithm uses the default parameters of the GAAssistant and GAAllocation objects of Plant Simulation. These include the number of generations G_{num}, the crossover rate p_c, the mutation rate p_m and the selection operator p_s. Chambers (2001) and Goldberg (1989) explain the theoretical principles of genetic algorithm operators and their impact on the optimization process while Rolf et al. (2020) described the implementation of the common operators in Plant Simulation in their previous work.

During the optimization the simulation model automatically saves the outputs of the iterations specified in the table of saving points. This step in the optimization process is referred as documentation in Figure 3 and also checks if the termination criterion is reached. The model saves not only the makespan, total tardiness and number of tardy jobs for every so far found solution but also a complete schedule for all machines that is ready for usage in the production system of the PCB company. The GanttWizard object of Plant Simulation can also convert the table to a schedule in the form of a Gantt chart quickly.

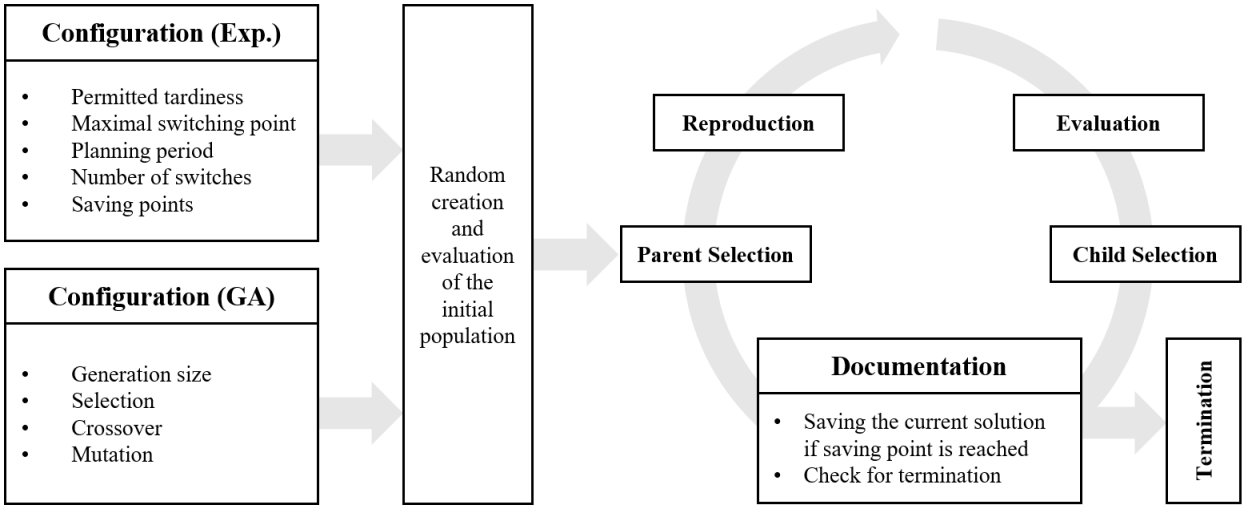

Figure 3: Flow chart of the optimization process of the simulation-based genetic algorithm.

4 COMPUTATIONAL RESULTS

We took the simulation-based optimization model presented in section 3 and used it on various experiment configurations and optimization tasks on real-world datasets of the PCB assembly company. All experiments were conducted on a PC with standard hardware (CPU: i5-4670, 4 x 3.40 GHz, RAM: 16 GB) using distributed simulation of Plant Simulation for running simulations on four kernels simultaneously. The four datasets contain jobs for a production period of three weeks with their product families, due dates and processing times for the SMT and AOI stages.

In this experiment we investigated the relation between the number of switching points, the quality of the obtained solutions in terms of the multicriteria objective function and computation time. Table 1 shows the experiment configuration and table 2 shows the configuration of the genetic algorithm. The experiments aim to optimize four real-world datasets of the PCB manufacturing company that have been studied intensively in the past in Rolf et al. (2020), Aurich et al. (2017) and Nahhas et al. (2016). Hence, the approximate outputs of the final solutions are known and the switching periods can be adapted to reduce the solution space. Tardy jobs are completely forbidden because the PCB company requires on-time delivery for all jobs.

The experiments were conducted with four different numbers of switching points $(i = 5, i = 10, i = 20, i = 50)$ for each data set resulting in 16 optimization runs. Table 3 shows the solutions after 400 generations with the corresponding makespan, total tardiness and number of major setups. The results make clear that a genetic algorithm with more genes respectively a higher number of switching points finds better solutions with respect to the objective function. The solutions also prove that few major setups are not necessarily needed to find a solution that has a minimal objective function value and fulfills the restrictions.

To show the relationship between the objective function value and the required computation time we exemplary analyze data set 1. Table 4 shows the number of needed genes in the representation and the

Table 1: Configuration of the experiment.

Parameter	Configuration (Experiment)
Permitted tardiness (min)	$T_{per} = 0$
Switching period (min)	$S_{period} = 17,000$ for dataset 1
	$S_{period} = 20,000$ for dataset 2
	$S_{period} = 19,000$ for dataset 3
	$S_{period} = 18,000$ for dataset 4
Length of the planning period (d)	$t = 21$
Switching points (number)	$i = 5$ (Setup 1)
	$i = 10$ (Setup 2)
	$i = 20$ (Setup 3)
	$i = 50$ (Setup 4)
Saving points (number)	$G_{save} = (25, 50, 100, 200, 400)$

Table 2: Configuration of the genetic algorithm.

Parameter	Configuration (Genetic Algorithm)
Number of generations	$G_{num} = 400$
Generation size	$G_{size} = 100$
Fitness function	$F(I) = C_{max} + max\{0; T - T_{per}\}$
Crossover rate	$p_c = 0.8$
Mutation rate	$p_m = 0.01$
Selection operator	$p_s = 1of2$

computation time to process 400 generations in the simulation model depending on the number of switching points. The number of switching points and the number of genes are in a direct relation to each other because each switching point requires one set with three genes and additionally two genes for the initial dispatching rule combination. Determining the relationship between the number of switching points and the computation time is more difficult because not all impacts are known. For our experiments the relationship in the tested interval seems to be roughly linear so whenever the number of switching points doubles the computation time also doubles. Knowing this, the user is able to create a configuration that is appropriate to the available hardware and time.

Figure 4 shows the development of the objective function value compared to the computation time for all four investigated switching point setups. Each graph represents a switching point setup and ends after the genetic algorithm successfully processed 400 generations. We decided to display the computation time on the abscissa instead of the generations because a single generation needs different amounts of time

Table 3: Solutions with makespan, total tardiness and number of major setups after 400 generations.

	Switching points (number)	5	10	20	50
Data set	Makespan (min)	16,705	16,467	16,314	16,223
1	Major setups (number)	94	71	72	67
Data set	Makespan (min)	20,118	19,724	19,722	19,581
2	Major setups (number)	102	118	116	114
Data set	Makespan (min)	18,677	18,425	18,285	18,085
3	Major setups (number)	104	101	93	85
Data set	Makespan (min)	17,567	17,526	17,600	17,400
4	Major setups (number)	78	74	90	84

Table 4: Relationship between number of switching points, number of genes and computation time.

Switching points (number)	5	10	20	50
Genes (number)	17	32	62	152
Computation time (min)	15	40	75	180

in each setup. Due to the real-world origin of the scheduling problem the required computation time to reach a certain level of solution quality is much more important than the required number of generations. The figure shows that each graph starts with great improvements in the first generations but converges to a limiting value after some time. The setups with less switching points are converging faster than the setups with more switching points but tend to converge to a lower objective function value. The results for the other data sets in table 3 confirm this observation. The setup with $i = 5$ switching points seems to be unsuitable for finding a schedule for this data set because it is dominated by the other graphs for every point in time. All other setups are suitable in certain intervals which are defined by the intersections of the graphs. Setup 2 and setup 3 intersect after approximately 20 minutes and setup 3 and setup 4 intersect after 100 minutes. Hence, a setup with $i = 10$ switching points is most effective when the available computing time is below 20 minutes while the second setup is most effective for situations with a mediocre time availability and the third setup for situations with a high availability of time. The results of the other data sets are nearly similar.

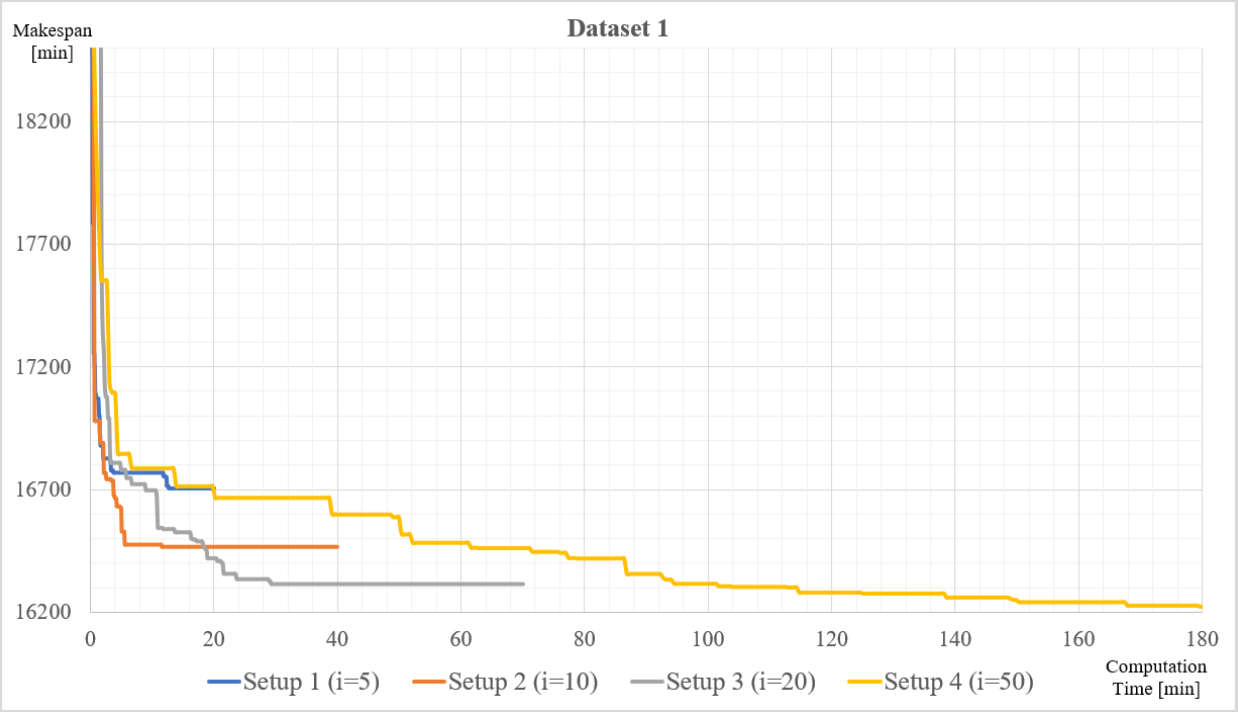

Figure 4: Computation time requirement for different switching point setups for data set 1.

Figure 5 illustrates the switching points and the sequence of dispatching rules of the best solution with $i = 20$ switching points. The bars represent the two production stages and both together make up a dispatching rule combination for each point in time. Each color represents one of the eight standard dispatching rules. The abscissa shows the time and its maximum value refers to the makespan $C_{max} = 16,314$ minutes of the shown solution. Although the switching period is limited ($S_{max} = 17,000$) the value of one switching point is higher than the makespan of this solution. Hence, figure 5 contains only 19 switching points even though 20 switching points were given by the experiment configuration. This cannot be

prevented in the current version of the simulation model but is not decisive for a good solution. Most of the eight dispatching rules are used frequently or at least once so this solution shows that our approach works as expected and every dispatching rule is superior to others in certain situations. Also the flexible determination of switching points through the genetic algorithm is successful as there are longer and shorter periods of rule assignment.

Figure 6 additionally shows the distribution of job-based and family-based dispatching rules for the same solution as in figure 5. Blue color represents job-based and orange color represents family-based dispatching rules. As stated in section 3 the genetic algorithm only assigns job-based dispatching rules to the AOI stage while the SMT stage uses both groups because of the sequence-dependent setup times. It is striking that job-based and family-based rules constantly alternate in the sequence for the SMT stage which is due to the contrary effects of both groups of dispatching rules. The family-based rules are particularly effective when multiple jobs of the same product family are currently in the buffer while the job-based rules are useful when the composition in the buffer is mixed. Checking the Gantt chart of the solution confirms that in the periods in which family-based rules control the dispatching the machines process families with a lot of jobs.

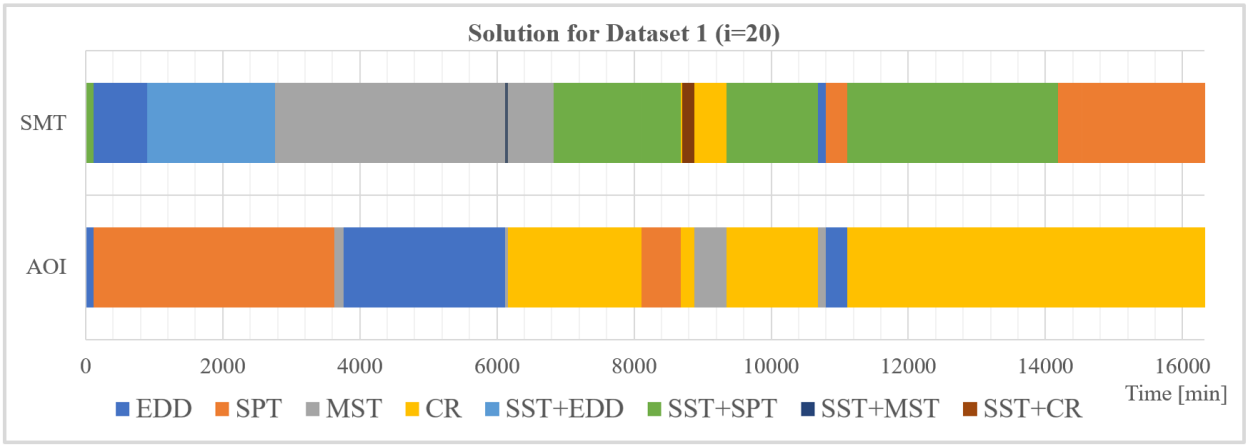

Figure 5: Distribution of dispatching rules with $i = 20$ switching points.

Figure 6: Distribution of job-based and family-based dispatching rules with $i = 20$ switching points.

The setup with $i = 5$ switching points of our dispatching rule-based approach generates already schedules of a similar quality compared to the solution approaches of Aurich (2017) and Nahhas et al. (2016). With

more than five switching points the quality of the schedules is visibly superior to their approaches. The makespan is 500-1500 minutes lower than in their best schedules for all four data sets. In terms of computation time all approaches are similar except the integrated simulation-based optimization (ISBO) approach of Nahhas et al. (2016) which can generate schedules in a few seconds. The ISBO is faster because it is specifically designated to the mentioned scheduling problem while all other approaches are adapted versions of general use metaheuristics.

The behavior of the different switching point setups also corresponds to the observations of Ochoa et al. (2009). They conducted experiments with dispatching rule-based representations of 10, 30 and 50 switching points as well as the permutation representation for instances of different sizes. In their experiments all dispatching rule-based representations outperformed the permutation representation and the representation with a medium number of switching points always performed best considering limited computation time. In our experiments the setups with $i = 5$ and $i = 50$ switching points also perform worse if a solution is needed quickly.

5 CONCLUSION AND FUTURE RESEARCH

The paper has shown that alternating standard dispatching rules with a genetic algorithm embedded in a simulation model can solve a complex real-world hybrid flow shop scheduling problem with sequence-dependent setup times. The simulation-based genetic algorithm found solutions that are feasible, fulfill the requirements of the company and achieve a good objective function value. The newly implemented flexible configuration of the simulation model is helpful to speed up the optimization process and to adapt it to the given data set.

The different switching point setups in figure 4 show that each setup is most effective during a certain time interval. An even more flexible optimization model could take the intersections of the graphs into account and adapt the configuration within the optimization run. For example, the model could automatically increase the number of switching points whenever the limiting value is nearly reached and use the current solution as the initial solution for the new configuration. In this approach the genetic algorithm would always stay at the bottom of the curves and would possibly find good schedules even faster. Implementing this is beyond the technical possibilities of the genetic algorithm library of Plant Simulation and needs a totally new implementation in a more flexible programming language.

REFERENCES

Aurich, P. 2017. *Simulationsbasierte Optimierung von Hybrid-Flow-Shop Scheduling-Problemen mit reihenfolgeabhängigen Rüstzeiten*. Masterarbeit, Otto-von-Guericke-Universität, Magdeburg.

Aurich, P., A. Nahhas, T. Reggelin, and M. Krist. 2017. "Simulation based optimization of a four stage hybrid flow shop with sequence-dependent setup times and availability constraints". In *Proceedings of the 16th International Conference on Modeling and Applied Simulation*, edited by A. G. Bruzzone, F. De Felice, C. Frydman, F. Longo, M. Massei, and A. Solis, 144–152: Curran Associates.

Baker, K. R., and D. Trietsch. 2009. *Principles of Sequencing and Scheduling*. Hoboken: John Wiley & Sons.

Banks, J., J. S. Carson, II, B. L. Nelson, and D. M. Nicol. 2010. *Discrete-Event System Simulation*. Upper Saddle River: Pearson Education.

Blackstone, J. H., D. T. Phillips, and G. L. Hogg. 1982. "A state-of-the-art survey of dispatching rules for manufacturing job shop operations". *International Journal of Production Research* 20(1):27–45.

Burke, E. K., M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. 2013. "Hyper-heuristics: a survey of the state of the art". *Journal of the Operational Research Society* 64(12):1695–1724.

Chambers, L. (Ed.) 2001. *The practical handbook of genetic algorithms: Applications*. 2. ed. ed. Boca Raton, Fla.: Chapman & Hall/CRC.

Cheng, R., M. Gen, and Y. Tsujimura. 1996. "A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms: I. Representation". *Computers & Industrial Engineering* 30(4):983–997.

Dorndorf, U., and E. Pesch. 1995. "Evolution Based Learning in a Job Shop Scheduling Environment". *Computer & Operations Research* 22(1):25–40.

Garey, M. R., and D. S. Johnson. 1996. *Computers and intractability: A guide to the theory of NP-completeness*. 27. Auflage ed. A series of books in the mathematical sciences. New York: Freeman.

Glass, C. A., and C. N. Potts. 1998. "Machine scheduling". In *Local Search in Combinatorial Optimization*, edited by E. H. L. Aarts and J. K. Lenstra, Wiley-Interscience series in discrete mathematics and optimization, 361–414. Chichester: John Wiley & Sons.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley.

Gosavi, A. 2015. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, Volume 55 of *Operations Research Computer Science Interface Series*. Boston: Springer.

Gupta, J. N. D. 1988. "Two-Stage, Hybrid Flowshop Scheduling Problem". *Journal of the Operational Research Society* 39(4):359–364.

Haupt, R. 1989. "A Survey of Priority Rule-based Scheduling". *OR Spektrum* 11(3):3–16.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: MIT Press.

Huang, J., and G. A. Süer. 2015. "A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels". *Computers & Industrial Engineering* 86:29–42.

Nahhas, A., P. Aurich, T. Reggelin, and J. Tolujew. 2016. "Heuristic and Metaheuristic Simulation-Based Optimization for Solving a Hybrid Flow Shop Scheduling Problem". In *Proceedings of the International Conference on Modeling and Applied Simulation*, edited by A. G. Bruzzone, F. De Felice, C. Frydman, M. Massei, Y. Merkuryev, and A. Solis, 95–103.

Ochoa, G., J. A. Vazquez-Rodriguez, S. Petrovic, and E. Burke. 2009. "Dispatching rules for production scheduling: A hyper-heuristic landscape analysis". In *IEEE Congress on Evolutionary Computation*, 1873–1880.

Pinedo, M. L. 2016. *Scheduling: Theory, Algorithms, and Systems*. Fifth Edition ed. Cham: Springer International Publishing.

Rolf, B., T. Reggelin, A. Nahhas, S. Lang, and M. Müller. 2020. "Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem". *Procedia Manufacturing* 42:442–449.

Talbi, E.-G. 2009. *Metaheuristics: From design to implementation*. Hoboken: John Wiley & Sons.

Vázquez-Rodríguez, J. A., and S. Petrovic. 2010. "A new dispatching rule based genetic algorithm for the multi-objective job shop problem". *Journal of Heuristics* 16(6):771–793.

Zhang, R., S. Song, and C. Wu. 2013. "A dispatching rule-based hybrid genetic algorithm focusing on non-delay schedules for the job shop scheduling problem". *The International Journal of Advanced Manufacturing Technology* 67(1-4):5–17.

AUTHOR BIOGRAPHIES

BENJAMIN ROLF is a master student and research assistant in the Institute of Logistics and Material Handling Systems at Otto von Guericke University Magdeburg. He holds a bachelor's degree in Industrial Engineering with specialization in Logistics from Otto von Guericke University Magdeburg. His research interests include modeling, simulation, optimization and applying methods of artificial intelligence for systems in production and logistics. His email address is benjamin.rolf@ovgu.de.

TOBIAS REGGELIN is a project manager, researcher and lecturer at Otto von Guericke University Magdeburg and Fraunhofer Institute for Factory Operation and Automation IFF Magdeburg. His main research and work interests include modeling and simulation of production and logistics systems and developing and applying logistics management games. Tobias Reggelin received a doctoral degree in engineering from Otto von Guericke University Magdeburg. Furthermore, he holds a master's degree in Engineering Management from Rose-Hulman Institute of Technology in Terre Haute, IN and a diploma degree in Industrial Engineering in Logistics from Otto von Guericke University Magdeburg. His email address is tobias.reggelin@ovgu.de.

ABDULRAHMAN NAHHAS is a research fellow at Otto von Guericke University Magdeburg. His main work interests are scheduling problems and simulation-based optimization solution strategies. He holds a master's degree in Business Informatics from Otto von Guericke University Magdeburg. His email address is abdulrahman.nahhas@ovgu.de.

MARCEL MÜLLER is a research fellow at Otto von Guericke University Magdeburg. He earned his master's degree in Industrial Engineering in Logistics at Otto von Guericke University Magdeburg. His research interests include modeling and simulation of logistics systems and handling of deadlocks. His email address is marcel1.mueller@ovgu.de. His website is https://www.ilm.ovgu.de/mueller.

SEBASTIAN LANG is a research fellow at Fraunhofer Institute for Factory Operation and Automation IFF. He holds a master's degree in mechanical engineering with focus on production technologies and a master's and bachelor's degree in industrial engineering and logistics. His research interests include studying and applying methods of machine learning and artificial intelligence, simulation modeling and mathematical optimization for problems in production and logistics. His e-mail address is sebastian.lang@iff.fraunhofer.de and he has the following profile on ResearchGate.