Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

LARGE-SCALE SIMULATION-BASED OPTIMIZATION OF SEMICONDUCTOR DISPATCHING RULES

Torsten Hildebrandt
Debkalpa Goswami

BIBA - Bremer Institut für Produktion und Logistik GmbH at the
Universität Bremen
Hochschulring 20
Bremen, 28359, GERMANY

Michael Freitag

ArcelorMittal Bremen

Models & Simulation
Carl-Benz-Straße 30
Bremen, 28237, GERMANY

ABSTRACT

Developing dispatching rules for complex production systems such as semiconductor manufacturing is an involved task usually performed manually. In a tedious trial-and-error process, a human expert attempts to improve existing rules, which are evaluated using discrete-event simulation. A significant improvement in this task can be achieved by coupling a discrete-event simulator with heuristic optimization algorithms. In this paper we show that this approach is feasible for large manufacturing scenarios as well, and it is also useful to quantify the value of information for the scheduling process. Using the objective of minimizing the mean cycle time of lots, we show that rules created automatically using Genetic Programming (GP) can clearly outperform standard rules. We compare their performance to manually developed rules from the literature.

1 INTRODUCTION

Scheduling semiconductor factories is a very challenging task due to the high complexity of the manufacturing process as well as the importance of this task in a competitive and capital-intensive production environment. This high complexity, and the requirement to be able to react quickly to disruptions on the shop floor (e.g. machine failures), have led to the prevalent use of dispatching rules (Pfund, Mason, and Fowler 2006). Dispatching rules can be used to take decisions in real-time. Besides their low computational requirements, they are easy to understand and implement, providing the flexibility to seamlessly incorporate domain knowledge and expertise (Aytug et al. 2005). Incorporating such knowledge in an effective way is still a tedious trial-and-error process (Geiger, Uzsoy, and Aytug 2006). It usually requires a large amount of expertise time and coding effort, to test, modify and potentially retest a candidate rule until results are satisfactory. To ease this process, in recent years, the use of hyper-heuristics (Burke et al. 2013) has been proposed. In this approach, a meta-heuristic is commonly used to select the most suitable heuristic for a problem at hand or to develop an entirely new heuristic. Automatically finding dispatching rules belongs to the latter category.

Building upon and refining work by Pickardt et al. (2010) and Branke, Hildebrandt, and Scholz-Reiter (2014), we investigate the use of this hyper-heuristic approach to find dispatching rules minimizing the mean cycle time of lots. For this purpose, the complex MIMAC FAB6 model (233 machines; process flows with up to 355 operations, more than 3000 jobs per simulation run) is employed. We can show the potential of our proposed approach using 3 settings from the literature, where rules were developed manually for this model.

.

This paper is structured as follows. After concisely reviewing the related work in Section 2, we present our approach in Section 3. Computational experiments are presented in Section 4 and results using a basic attribute set are discussed in Sections 5. In Section 6 we investigate, how results are effected, if operational due dates are available as additional information in dispatching rules. Section 7 more closely analyses rule performance in terms of mean cycle time, mean tardiness, and average work in processes by comparing the results of the best GP-evolved rules with the best benchmark rule. The paper concludes with a summary and outlook towards future work in Section 8.

2 PREVIOUS WORK

Several authors have used Genetic Programming (GP) (Koza 1992; Poli et al. 2008) to generate dispatching rules that assemble basic job, machine and system attributes into priority indices. To name just a few, Dimopoulos and Zalzala (2001), Geiger, Uzsoy, and Aytug (2006) and Geiger and Uzsoy (2008) test this approach on various single machine problems. Jakobović and Budin (2006) investigate the use of GP for job shops. Tay and Ho (2008) do the same on a flexible job shop. The authors report that evolved rules are usually able to achieve better results than standard rules.

There is only very little work on using GP to evolve dispatching rules for more complex scheduling settings. Pickardt et al. (2010) used the MIMAC FAB4 model to find dispatching rules minimizing weighted tardiness. Their very encouraging results were further improved upon in follow-up works (Pickardt et al. 2013; Pickardt 2014) using a two-stage approach. In this approach, the first stage uses GP to find a good dispatching rule, and the second stage can improve on GP's results by choosing individually for each machine whether the evolved rule or a rule from a pre-defined set of standard rules is supposed to be used. Using the two-stage approach, mean weighted tardiness of jobs can further be improved. Pickardt (2014) also presents results using GP to evolve dispatching rules for a simplified version of the FAB6 model not considering downtimes. Evolved rules outperform standard benchmark rules in terms of cycle time and mean weighted tardiness.

The simulation model used in this paper is from the MIMAC (Measurement and Improvement of Manufacturing Capacities) testbed (Fowler and Robinson 1995; Feigin, Fowler, and Leachman 1996). This dataset was used in a number of research activities to improve scheduling and dispatching of complex manufacturing facilities, especially to manually develop dispatching rules. In Zhou and Rose (2011) a composite dispatching rule is presented as a linear combination of ODD (operational due date), SPT (shortest processing time next), and LWNQ (least work in next queue). Their combined rule with proper weight values is reported to outperform the MOD rule with regard to cycle time as well as tardiness-related objectives. Zhou and Rose (2012) use the FAB6 model to present extensions to the IFD rule (see section 4.2). Their extended rule, incorporating ideas to balance work in process and selectively accelerate urgent lots, is reported to outperform the plain IFD rule. The primary concern of their paper is to improve tardiness-related objectives, but also improved cycle time results are reported.

3 APPROACH

Our general approach can be classified as using GP (Koza 1992; Poli et al. 2008) as a hyper-heuristic (Burke et al. 2013) to develop an entirely new heuristic in the form of a dispatching rule. GP is a meta-heuristic from the family of evolutionary algorithms that can be used to optimize tree-like data structures of variable length, such as the arithmetic expression of a dispatching rule. The advantages of this approach in the semiconductor scheduling context are as follows (Pickardt et al. 2010):

1. The time-consuming GP run can be made off-line. Once a good rule is found it can be used as an efficient, on-line/real-time scheduling rule just like any standard rule.
2. As an automated approach, very little manual work has to be invested, especially to:
   (a) adopt to different objective functions,
   (b) incorporate additional information or assess usefulness of such information for scheduling.
3. New dispatching rules can be easily integrated in existing software for production control and manufacturing simulation.

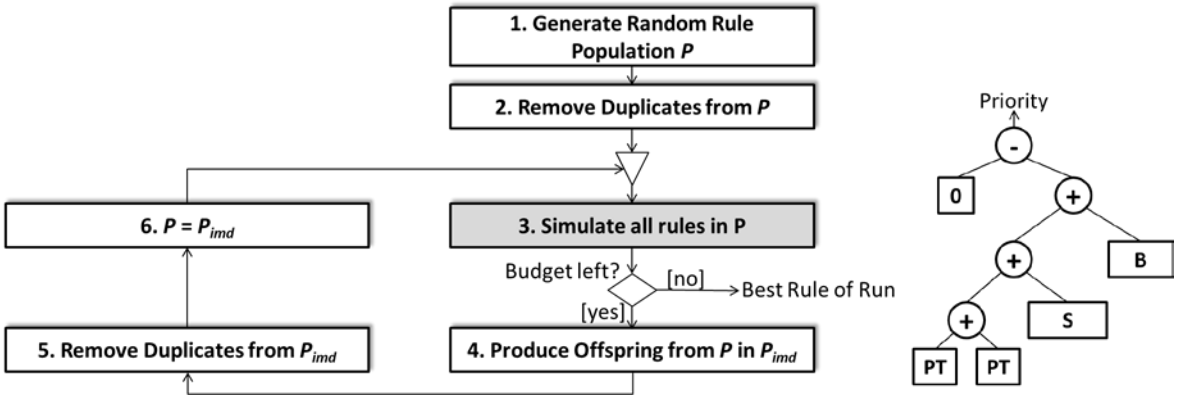

Figure 1: General solution approach. Left: simulation-based GP cycle; Right: sample GP tree encoding a hypothetical dispatching rule "0-(PT+PT+S+B)".

The approach to automatically generate rules is shown graphically on the left hand side of Figure 1. It depicts the general cycle of simulation-based GP. The algorithm starts with a population, P, of initially random dispatching rules. In step 2, we remove probable duplicates (without using computationally expensive simulation runs) using the procedure described by Branke, Hildebrandt, and Scholz-Reiter (2014). In step 3, all rules in this population are evaluated using discrete event simulation. In a simulation-optimization context as used here, this step is by far the most computationally demanding. As we use a population-based optimization algorithm, we always have to evaluate a large number of rules in this step. All simulations for a certain rule are independent of each other, so this step can easily be parallelized to take advantage of modern multi-core processors. Should the simulation time still be too high, distributing the simulation runs in a cloud-like environment would also be possible.

Once the performance of each rule is known, this information is used in step 4 to create the next generation using selection, cross-over and mutation operators to create new rules. GP works on a tree representation; so each individual (i.e. a dispatching rule) is a tree structure, which can get arbitrarily complex. A sample tree encoding the arithmetic expression "0-(PT+PT+S+B)" is shown on the right hand side of Figure 1. GP's cross-over and mutation operators work on such tree-structures, combining individuals to form new ones, and randomly changing/mutating parts of a tree. How such a tree should look like, i.e., what can be used as inner nodes (arithmetic operations) and what information can be used as leaf nodes (constants and a set of scheduling-relevant information), is a decision the user has to make before the optimization run.

Step 5 again is similar to step 2 as it uses the procedure from Branke, Hildebrandt, and Scholz-Reiter (2014) to detect and remove potential duplicates, without having to simulate such rules. In step 6, the new population replaces the old one and the main optimization cycle is repeated, starting with step 3. The algorithm finally terminates if a certain pre-specified computational budget, i.e. a certain number of simulation runs (we use 30,000 for the experiments of this paper), has been expended. The best rule is returned as the result of the optimization run.

4 EXPERIMENTAL SETUP

4.1 Simulation Model

For the experiments in this paper, we use the FAB6 model from the publically available MIMAC (Measurement and Improvement of Manufacturing Capacities) testbed (Fowler and Robinson 1995; Feigin, Fowler, and Leachman 1996). This model represents a semiconductor manufacturing facility with the following characteristics:

- 104 tool groups with a total number of 223 machines,

- Batch machines; sequence-dependent setup times,
- 9 process flows/products, having between 234 and 355 operations,
- Machine downtimes (failures and maintenance).

Depending on the setting (see Table 1), between 3143 and 3795 lots are started during the 18-month period used for a simulation run.

The FAB6 model has been used previously in research to manually develop dispatching rules, and therefore offers good benchmarks. We use three different settings from the literature, as summarized in Table 1. The first setting is based on Zhou and Rose (2011). The second and third settings are based on Zhou and Rose (2012). Following their setup for each of these settings, we simulate the system for a total duration of 18 months, ignoring data from the first 6 months to focus on the system's steady state behavior. It must be noted that the start rates are slightly lower than the original rates, i.e. the inter-lot release time given in Table 1 is higher than the original values. This is because the simulator used by Zhou and Rose (2011; 2012) uses the scrap setting present in the original FAB6 model. As this feature is currently not implemented in our simulation library, we instead decreased the start rates while keeping the product mix constant so the overall bottleneck utilization remains the same.

We implemented this model in the *jasima* simulation library (*jasima* - an efficient Java Simulator for Manufacturing and Logistics; http://jasima.googlecode.com). *jasima* has been used before in the context of simulation-based optimization to select or create dispatching rules for job shops (Pickardt and Branke 2012; Branke, Hildebrandt, and Scholz-Reiter 2014) or semiconductor scenarios (Pickardt et al. 2010; Pickardt et al. 2013; Pickardt 2014).

The FAB6 model includes random influences in the form of machine downtimes. Both the duration and time between downtimes are given using an exponential distribution. All other model parameters, especially job arrivals, are deterministic.

Table 1: Start rates and flow factors used.

Product	Setting 1, 95%		Setting 2, 99.5%		Setting 3, 99.4%	
	New lot released every … hours	Flow Factor	New lot released every … hours	Flow Factor	New lot released every … hours	Flow Factor
C6N3	44.60	2.30	86.71	1.80	29.07	2.40
B6HF	100.90		173.05	1.80	26.44	2.40
C4PH	47.67		86.71	1.80	14.53	2.40
C6N2	51.72		41.61	2.40	43.59	2.20
OX2	38.28		41.61	2.40	43.59	2.20
C5F	39.53		41.61	2.40	43.59	2.20
C5P	11.86		11.89	2.60	90.81	1.30
C5PA	18.70		13.87	2.60	90.81	1.30
B5C	33.07		27.75	2.60	90.81	1.30

4.2 Benchmark Rules

In order to assess the performance of our GP approach, we first conducted a normal simulation study to find the best combination of (standard) dispatching rule and batching policy. As dispatching rules we consider the standard rules (Haupt 1989) FIFO (first in first out), ERD (earliest release time first), EDD (earliest due date first), ODD (operational due date), MOD (modified operational due date), SPT (shortest processing time first), and CR (critical ratio). Additionally we also test the IFD rule as given in Zhou and Rose (2012). This rule is mentioned there to be in use at Infineon Technologies AG Dresden and divides waiting lots in 3 categories: highest priority is given to jobs already waiting longer than 2 days, medium priority to all other jobs that are already late according to their operational due dates, and all other lots are

put into the lowest priority class. A tie breaker rule is used to distinguish between jobs within the same class. For this purpose we use the ODD rule as suggested in Zhou and Rose (2012).

All rules are used with the ERD rule as a final tie breaker. As some machines require setup times, all of these rules are used with a setup-avoidance strategy, improving cycle times considerably. Setup time information for the GP-evolved rules is incorporated directly into the arithmetic expression of the rule.

To control batch formation we use greedy batching with a certain minimum batch size (MBS). Batch formation using MBS first sequences waiting jobs using one of the sequencing rules just described. Then the job with the highest overall priority determines the family of the formed batch. This batch is filled with as many compatible jobs as possible up to the maximum batch size. As a parameter in this heuristic we use a minimum batch size to enforce a certain utilization of machine capacity. In the experiments below we denote this using the abbreviation MBS(u) where u is a number between 0 and 1 to specify the minimum batch size as a fraction of the maximum size allowed.

In the preliminary study to find the best standard rule setting we considered all combinations of 6 settings for the batching policy (MBS(0), MBS(0.2), MBS(0.4), MBS(0.6), MBS(0.8), MBS(1)) and the 8 dispatching rules mentioned above.

4.3 Genetic Programming Settings

The simulation was coupled with the Evolutionary Computation in Java (ECJ) framework (version 20, available from http://cs.gmu.edu/~eclab/projects/ecj/, accessed June 06, 2014) for performing the GP operations. It uses the standard sub-tree crossover (produces two offspring trees by exchanging a randomly selected sub-tree between the two parents) and point mutation (replacing a node of the tree by a new, randomly created sub-tree) as described by Koza (1992). The ramped half and half initialization method ensures that the generated trees have a variety of sizes and shapes. The detailed settings used for GP are summarized in Tables 2 and 3.

Table 2: GP parameters used.

Name	Value
Computational Budget	30,000 simulations (60 generations)
Population Size	500
Crossover Proportion	90%
Mutation Proportion	10%
Elitism	10 individuals
Selection Method	Tournament Selection (size 7)
Creation Type	Ramped Half-and-Half (Min. Depth 2, Max. Depth 6)
Max. Depth for Crossover	17
Operators	+, -, *, /, max, if-then-else
Terminals	Constants: 0, 1, 2, 5; 8 attributes, see Table 3

Besides these settings, we use the removal of (probably) equivalent rules as well as the normalization of attributes into a common value range of (0,2). It has been shown by Branke, Hildebrandt, and Scholz-Reiter (2014) that both of these settings improve convergence speed and reduce variability of optimization results of GP in the case of finding dispatching rules for a dynamic job shop scheduling problem. Some of the normalization ranges used (see Table 3) are straightforward to obtain (e.g., “pt” or “s”), but some, such as “tis” and “tiq”, are more difficult, or theoretically even unlimited. To obtain reasonable normalization ranges for these attributes, we performed preliminary simulation runs using the FIFO rule and used estimated 95% percentiles as the upper limit of the normalization range.

To account for the randomness in the simulation model, we used a single replication to evaluate the individuals during a GP run. We, however, change the random number seed after each generation. Doing

so prevents GP from finding solutions optimized for a particular setting caused by a certain seed, while still using common random numbers to rank all individuals of a certain generation. Therefore 60 seeds, i.e. problem instances, are used during an optimization run. Performance figures in Section 5 are averaged over a different set of 30 problem instances.

Table 3: Attributes usable in evolved dispatching rules.

Attribute	Norm. Range	Explanation
pt	[7,1470]	Processing time of a job's current operation
ptAvg	[7,1470]	Average processing time of all waiting jobs
s	[0,90]	Setup time
sAvg	[0,90]	Average setup time of all waiting jobs
rOps	[1,355]	Number of remaining operations
tis	[0,40800]	Time in System
tiq	[0,530]	Time in Queue
b	[1,18]	Batch family size. How many compatible jobs are in queue?

5 RESULTS FOR BASIC ATTRIBUTE SET

At first, we performed 10 optimization runs for each of the 3 settings as given in Table 1using the GP settings as outlined in the previous section. The upper part of Table 4 shows the mean cycle times obtained in days. For each setting, we give the results for a default FIFO rule, the best results obtained by any of the benchmark rules tested, as well as average performance (GP_{avg}) and best performance (GP_{best}) of the rules obtained using GP. All results for individual rules are averaged over 30 independent replications, with twice the standard error shown in brackets as a measure of result uncertainty. As an exception to this, the row "GP_{avg}" shows the average performance of the 10 GP-evolved rules for each setting with uncertainty information showing twice the standard error across these 10 results. Finally, the reduction in average cycle time between GP_{best} and FIFO and between GP_{best} and the best benchmark rules are given.

As far as the selection of the best benchmark rule is concerned, choosing no minimum batch size MBS(0) seems to be advisable for all three scenarios considered. This is probably due to the high system utilization which is already sufficient to ensure a high enough batch utilization. Among the standard rules ODD and MOD give good results, but are usually outperformed slightly by IFD. Selecting a proper dispatching rule yields a cycle time reduction between 1.0 and 1.6 days compared to FIFO. This can be significantly improved upon using the GP approach. Evolved rules yield a reduction of cycle times between 2.1 and 4.5 days, just by using standard information which is also used in standard rules in a more efficient way. Comparing to FIFO, this is a relative reduction of 12.6 %, 14.4 %, or 8.3 % in average cycle times. This is a reduction in its reducible component, i.e., the sum of waiting and setup times, of 23.8 %, 25.6 % and 17.8 % respectively. Improvements over the best standard rule are lower, but still very significant.

Comparing results with those from the literature turned out to be difficult, but is attempted in the lower part of Table 4. We try to compare indirectly, that is, without re-implementing their methods/rules in our simulation software. As mentioned in section 4.1, our software currently does not support the "scrap" feature. This feature has a significant effect on cycle time results for the FAB6 model, even if we compensate for the lower capacity available when scrap is not considered. To have some baseline results, we used the Factory Explorer software from Wright Williams & Kelly, Inc. to run simulations with the FIFO and ODD rules for all 3 settings, resembling the settings used in Zhou and Rose (2011; 2012) as closely as possible. This means that scrap was enabled and a single replication using the default random number seed was performed. Results in the row "manual rule" were taken from Zhou and Rose's (2011) "composite rule" for Setting 1 and Zhou and Rose's (2012) "IFD+W+D"-rule for Settings 2 and 3.

A detailed comparison with our GP results is difficult, but judging from the improvements compared to FIFO, there seems to be an advantage of the rules found by GP over the manual rules. This does not

seem to be the case for Setting 3. Enabling/disabling scrap has a particularly large effect for this setting and the manually developed rule can reduce mean cycle time compared to FIFO by 3.6 days. A closer investigation is needed to find out why the manually developed "IFD+W+D"-rule achieved a better performance for Setting 3, but not for Setting 2. Maybe this can help to either further improve the manual rule or find an additional attribute that helps GP to find better rules for Setting 3 as well.

Table 4: Cycle time results. All times are in days.

	Setting 1		**Setting 2**		**Setting 3**	
	optimization results					
FIFO result:	28.6 (±0.13)	FIFO;MBS(0)	31.2 (±0.23)	FIFO;MBS(0)	25.3 (±0.12)	FIFO;MBS(0)
best benchmark:	27.1 (±0.10)	IFD;MBS(0)	29.6 (±0.20)	IFD;MBS(0)	24.3 (±0.10)	IFD;MBS(0)
GP_{avg}:	25.3 (±0.12)		26.9 (±0.15)		23.4 (±0.14)	
GP_{best}:	25.0 (±0.08)		26.7 (±0.15)		23.2 (±0.08)	
improvement GP_{best} over FIFO:	3.6 days		4.5 days		2.1 days	
impr. GP_{best} over best benchmark:	2.1 days		2.9 days		1.1 days	
	results from literature					
FIFO result:	29.8		33.2		28.5	
ODD result:	27.2		29.9		27.1	
manual rule:	27.9		30.0		24.9	
improvement over FIFO:	1.9 days		3.2 days		3.6 days	

We also investigated the robustness of the GP-evolved rules. If rules are very specifically adopted to a certain setting of parameters, they might show only poor performance when applied to new or different situations. To check this, we applied each of the 10 rules optimized for Setting 1(, Setting 2, Setting 3), denoted by Rules1(, Rules2, Rules3), to all scenarios. Results are shown in Table 5. As expected, best results are obtained when applying the rules to the setting they were developed for. Especially Settings 1 and 2 seem to be similar and rules developed for either of the scenarios work well for the other as well. This is probably due to the fact that basically the same product mix is used for both settings, only utilization is increased in Setting 2. Rules developed for Setting 3 work best in this setting and show only a weak performance for Settings 1 and 2. They can still achieve a better performance than the best benchmark rule even for these settings.

Table 5: Cycle times achieved on applying GP rules to different settings. All times are in days.

	Setting 1	**Setting 2**	**Setting 3**
Rules1	**25.26 (±0.12)**	27.02 (±0.15)	23.68 (±0.13)
Rules2	25.28 (±0.11)	**26.92 (±0.15)**	23.89 (±0.20)
Rules3	25.80 (±0.13)	28.30 (±0.54)	**23.42 (±0.14)**

6 ADDITIONAL INFORMATION

Looking at the results in Table 4 again, we see that the IFD rule rules yield the best results Also the ODD, and MOD rules lead to good results, but are usually outperformed by IFD. All of these rules use operational due dates as something like intermediate milestones for the process steps. Even if there are no externally set due dates and the primary concern is reducing the cycle time of lots, it makes sense to use

these rules similar to a parameterized dispatching rule by carefully adjusting the flow factor of lots. This way, very good cycle times can usually be achieved.

As mentioned in Section 3, one advantage of our approach is that it can be used to assess the value or usefulness of certain information for scheduling decisions. We demonstrate this by adding two additional attributes to our set: the operational due date and the average operational due date of all jobs in the queue. Now, by performing 10 optimization runs again, we can compare results with our previous findings.

Table 6: Optimization results with different attribute sets.
All times are in days.

		Cycle Time [days]	
	Attribute Set	GP_{avg}	GP_{best}
Setting 1	old set	25.3 (±0.12)	25.0 (±0.08)
	incl. odd+avgOdd	24.9 (±0.05)	24.8 (±0.07)
	difference:	0.34	0.24
Setting 2	old set	26.9 (±0.15)	26.7 (±0.15)
	incl. odd+avgOdd	26.6 (±0.07)	26.4 (±0.13)
	difference:	0.36	0.26
Setting 3	old set	23.4 (±0.14)	23.2 (±0.08)
	incl. odd+avgOdd	23.1 (±0.05)	22.9 (±0.09)
	difference:	0.32	0.27

Results shown in Table 6 are in the same format as in Table 4, i.e., the numbers in brackets list twice the standard error; across 10 rules for GP_{avg}, and across 30 independent replications of the individual rules GP_{best}. Comparison of results shows that using the extended attribute set is preferable over our initial choice. Even though the search space for GP gets larger, it is able to find better solutions. The best rule found improves the cycle time over the old best rule by on average 0.26 days in each of the 3 settings investigated. There is an even larger improvement of average rule performance: 0.34 days. Last but not least, the variance for GP_{avg} clearly decreases for all three settings, which means, GP can find very good rules more reliably using the extended attribute set.

7 DIFFERENT OBJECTIVE FUNCTIONS

In the previous sections we only looked at the mean cycle time of lots. To assess the usefulness of a certain dispatching rule it is beneficial to also investigate its performance for other common objective functions as well. We therefore analyzed the performance of the best benchmark rule (IFD) with the best rule found by GP for each setting for 3 different objective functions. Results for mean cycle time (in days), mean tardiness (in hours) and average work in process (WIP, in number of wafers) are shown in Table 7.

The table again shows the very good performance of GP rules for mean cycle time, which is the objective function used during the optimization process. These rules do not just perform well for this objective function, however. Results for mean tardiness are slightly worse than IFD's performance for two out of three settings, but differences are not very high. For the third objective function investigated, average work in process, optimized rules again show a considerably better performance than IFD, reducing WIP by 8.5 %, 10.8 %, and 5.5 %, depending on the scenario. Therefore the slightly better performance of IFD regarding mean tardiness of jobs is accomplished by a considerably higher WIP level and an increase in cycle time by between 6.1 %, and 12.1 %.

Table 7: Detailed comparison with IFD rule.

		Cycle Time [d]	Tardiness [h]	WIP [wafer]
Setting 1	$GP_{best/odd}$	24.8 (±0.07)	0.0 (±0.00)	4168 (±12)
	IFD	27.1 (±0.10)	0.0 (±0.00)	4553 (±17)
	diff. IFD-GP:	2.29	0.00	385
Setting 2	$GP_{best/odd}$	26.4 (±0.13)	0.2 (±0.04)	4465 (±22)
	IFD	29.6 (±0.20)	0.0 (±0.01)	5008 (±33)
	diff. IFD-GP:	3.22	-0.18	543
Setting 3	$GP_{best/odd}$	22.9 (±0.09)	0.9 (±0.04)	3216 (±12)
	IFD	24.3 (±0.10)	0.4 (±0.02)	3403 (±15)
	diff. IFD-GP:	1.34	-0.56	187

8 CONCLUSION AND OUTLOOK

This paper used Genetic Programming (GP) to automatically find dispatching rules minimizing the mean cycle time of lots for a complex simulated semiconductor manufacturing facility with 223 machines, and up to 355 operations per process flow. Using up to 30,000 simulation runs in an optimization run we were able to find dispatching rules clearly outperforming benchmark rules from the literature. We apply our approach to 3 different settings taken from the literature.

In comparison with FIFO, average cycle time could be reduced by up to 14.4 % (4.5 days), which is a reduction in its reducible component, i.e. the sum of waiting and setup times, of up to 25.6 %.

We further demonstrate the usefulness of our approach to assess the value of certain information for the scheduling process. Motivated by the good mean cycle time performance of standard rules using the operational due date information, we perform additional optimization runs with an extended attribute set. By allowing GP to use this information in a dispatching rule, an improvement of 0.34 days in the average performance of optimized rules is achieved.

Also a more detailed comparison of rules performance between GP-evolved rules and the best benchmark, IFD, was performed. For other objective functions than those used in the optimization process GP rules showed slightly worse mean tardiness results for 2 of the 3 settings investigated. This good performance of IFD is achieved at the cost of a considerably higher work in process level (5.5 % to 10.8 % more wafers/lots) and higher mean cycle times (6,1 % to 12,1 % increase).

Comparing the performance of GP-evolved rules with manually developed rules for the same scenario appeared difficult and results were inconclusive. This certainly needs closer investigation in future work. Ideally, comparing and understanding the differences between manual and automatically developed rules can be beneficial to initiate an improvement process for better dispatching decisions.

As further future work, we are currently interested in applying our method and software implementation to even larger, more realistic scenarios. We are therefore eager to collaborate with semiconductor manufacturers offering real-life scenarios and tough benchmark dispatching rules.

Finally it would be interesting to extend the optimization algorithm. For instance it would be very interesting to improve the convergence speed of GP. Therefore, e.g., surrogate functions could be used to replace some of the computationally expensive simulation runs. An approach how to achieve this is presented in Hildebrandt and Branke (2014), extending it to more complex scenarios, such as the FAB6 model used here, is pending. Furthermore it would be very interesting to extend the GP approach to consider multiple objectives simultaneously during a single optimization run. The result of such an optimization run is a set of Pareto-optimal (non-dominated) dispatching rules, where each rule from this set offers a certain trade-off between, e.g., mean cycle time and mean tardiness. Given such a set of rules the user could finally decide which rule offers the best trade-off according to his preferences — based on quantitative data what trade-offs between conflicting objectives are possible for the system under consideration.

REFERENCES

Aytug, H., M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy. 2005. "Executing production schedules in the face of uncertainties: a review and some future directions." *European Journal of Operational Research* 161:86–110.

Branke, J., T. Hildebrandt, and B. Scholz-Reiter. 2014. "Hyper-heuristic Evolution of Dispatching Rules: A Comparison of Rule Representations." *Evolutionary Computation*. Accepted for publication.

Burke, E.K., M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan and R. Qu. 2013. "Hyper-heuristics: A Survey of the State of the Art." *Journal of the Operational Research Society*.

Dimopoulos, C., and A.M.S. Zalzala. 2001. "Investigating the use of genetic programming for a classic one-machine scheduling problem." *Advances in Engineering Software* 32:489–498.

Feigin, G., J. Fowler, and R. Leachman. 1996. "MASM test data sets." Available from http://masmlab.engineering.asu.edu/ftp.htm. Accessed June 06, 2014.

Fowler, J., and J. Robinson. 1995. "Measurement and improvement of manufacturing capacities (MIMAC): Final report." Technical Report 95062861A-TR, SEMATECH, Austin, TX.

Geiger, C. D., and R. Uzsoy, 2008. "Learning effective dispatching rules for batch processor scheduling." *International Journal of Production Research* 46:1431-1454.

Geiger, C. D., R. Uzsoy, and H. Aytug. 2006. "Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach." *Journal of Scheduling* 9:7–34.

Haupt, R. 1989. "A survey of priority rule-based scheduling." *OR Spektrum* 11:3–16.

Hildebrandt, T., and J. Branke. 2014. "On Using Surrogates with Genetic Programming." *Evolutionary Computation*. Accepted for publication.

Jakobović, D. and L. Budin. 2006. "Dynamic Scheduling with Genetic Programming." In *Proceedings of the 9th European Conference, EuroGP 2006*, 73-84. DOI: 10.1007/11729976_7.

Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Pfund, M. E., S. J. Mason, and J.W. Fowler. 2006. "Semiconductor manufacturing scheduling and dispatching: state of the art and survey of needs." In *Handbook of Production Scheduling*, edited by J. W. Herrmann, 213–241. New York: Springer.

Pickardt, C., J. Branke, T. Hildebrandt, J. Heger, and B. Scholz-Reiter. 2010. "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness." In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 2504–2515. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pickardt, C. W. and J. Branke. 2012. "Setup-oriented dispatching rules — a survey." *International Journal of Production Research* 50:5823–5842.

Pickardt, C. W., T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter. 2013. "Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems." *International Journal of Production Economics* 145:67–77.

Pickardt, C.W. 2014. "Evolutionary Methods for the Design of Dispatching Rules for Complex and Dynamic Scheduling Problems." PhD thesis, Warwick Business School, UK.

Poli, R., W.B. Langdon, and N.F. McPhee. 2008. *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. Accessed June 06, 2014.

Tay, J. C., Ho, N. B. 2008. "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems." *Computers & Industrial Engineering* 54:453-473.

Zhou, Z., and O. Rose. 2011. "A composite rule combining due date control and WIP balance in a wafer fab." In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 2085-2092. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zhou, Z., and O. Rose. 2012. "WIP balance and due date control in a wafer fab with low and high volume products." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J.

Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, 2019-2026. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

TORSTEN HILDEBRANDT is research associate at the BIBA - Bremer Institut für Produktion und Logistik GmbH at the University of Bremen, Germany. His research interests include simulation-based optimization, Evolutionary Algorithms, and hyper-heuristics. He currently finishes his PhD on the topic of automatic, efficient generation of dispatching rules for complex manufacturing systems. His email address is hil@biba.uni-bremen.de.

DEBKALPA GOSWAMI is visiting student researcher at Bremer Institut für Produktion und Logistik GmbH (BIBA), on leave from Jadavpur University, India, where he is a senior undergraduate student of Production Engineering. His email address is gos@biba.uni-bremen.de.

MICHAEL FREITAG, PhD, is project engineer at ArcelorMittal in Bremen. His activities include simulation-based analysis of manufacturing and logistic systems, optimization modeling for production scheduling, and developing planning and control systems for logistic processes. His email address is michael.freitag@arcelormittal.com.