Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

GENERATING DISPATCHING RULES FOR SEMICONDUCTOR MANUFACTURING TO MINIMIZE WEIGHTED TARDINESS

Christoph Pickardt
Jürgen Branke

Warwick Business School
The University of Warwick
Coventry, CV4 7AL, United Kingdom

Torsten Hildebrandt
Jens Heger
Bernd Scholz-Reiter

Bremen Institute of Production and Logistics – BIBA
University of Bremen
28359 Bremen, Germany

ABSTRACT

Dispatching rules play an important role especially in semiconductor manufacturing scheduling, because these fabrication facilities are characterized by high complexity and dynamics. The process of developing and adapting dispatching rules is currently a tedious, largely manual task. Coupling Genetic Programming (GP), a global optimization meta-heuristic from the family of Evolutionary Algorithms, with a stochastic discrete event simulation of a complex manufacturing system we are able to automatically generate dispatching rules for a scenario from semiconductor manufacturing. Evolved dispatching rules clearly outperform manually developed rules from literature.

1 INTRODUCTION

This work addresses the semiconductor production scheduling problem, a complex variant of the job shop scheduling problem, which is well-known to be NP-hard (Pfund, Mason, and Fowler 2006). For a detailed discussion of the classic job shop problem and its constraints the reader is referred to, e.g., (Haupt 1989).

In semiconductor manufacturing, several characteristics that go beyond the classic job shop problem further complicate scheduling. These include, but are not limited to (Uzsoy, Lee, and Martin-Vega 1992, Pfund, Mason, and Fowler 2006):

- Reentrant product flows, i.e., jobs visit a work center/machine more than once,
- Sequence-dependent setup times,
- Batch processing capabilities, i.e., machines that can simultaneously process more than one job at a time,
- Multiple machines of the same type operating in parallel in some of the work centers.

Because of these additional issues, the application of exact solution methods appears to be only sensible for very small problem-sizes (Mason et al. 2005). Furthermore, most real-world applications involve stochastic shop floor disruptions (e.g., machine failures) and dynamic environments (e.g., new jobs arriving over time), which would require a frequent re-optimization. Moreover, repeatedly solving the static problem to optimality based on current information does not imply optimality for the dynamic problem, as future events are likely to render previously optimal decisions sub-optimal (Baker 1977, Branke and Mattfeld 2005).

A common alternative to global scheduling are dispatching rules: Whenever a machine becomes available for processing another job, a simple priority rule is used to determine the next job to be processed on this machine, based on local information of the waiting jobs, such as processing time or due date. Because dispatching rules make scheduling decisions locally at the last possible moment in time, they naturally take into account any disruptions or changes in the shop floor. Other frequently mentioned beneficial properties of dispatching rules include their simple and intuitive nature, their ease of implementation within practical settings, and their flexibility to incorporate domain knowledge and

expertise. All these factors have led to their prevalent use in various industries including semiconductor manufacturing (Pfund, Mason, and Fowler 2006).

The major drawback of dispatching rules is that they lack a global view of the problem, i.e., they approach the overall scheduling problem by taking independent scheduling decisions based on the current, local conditions at the particular machine without consideration of the negative effects they might have on future decisions and on the overall objective function value. Although some dispatching rules have shown to perform well in a wide range of manufacturing environments, no single rule has yet been found to outperform all other rules for different shop configurations, operating conditions, or across various objectives (Blackstone, Phillips, and Hogg 1982, Haupt 1989, Rajendran and Holthaus 1999).

The development of customized dispatching rules is usually a tedious procedure requiring a significant amount of expertise, coding effort and time. The challenge is to design local, decentralized rules which result in a good global performance of a complex production environment. Generally, this is achieved by a trial-and-error procedure, with candidate rules tested in a simulation model of the considered manufacturing system, modified, and retested until they fulfill the requirements for actual implementation in practice (Geiger, Uzsoy, and Aytug 2006).

In this paper, we use Genetic Programming (GP) combined with discrete event simulation to automatically generate dispatching rules for a semiconductor manufacturing environment. An empirical comparison with a number of state-of-the-art dispatching rules from the literature demonstrates the superiority and robustness of our automatically generated rules.

This paper is organized as follows. The proposed approach and related literature is presented in Section 2. Section 3 details the experimental design and results are presented in Section 4. Section 5 investigates the robustness of the developed dispatching rules, investigating how they perform under varied conditions. Finally, conclusions and future directions are given in Section 6.

2 SIMULATION-BASED GENETIC PROGRAMMING FOR THE GENERATION OF DISPATCHING RULES

Evolutionary Algorithms (EAs) are iterative metaheuristics inspired by natural evolution. They work with a set (population) of solutions (individuals), and, in each iteration, select good solutions from the population (survival of the fittest), and generate new solutions by recombining the information of two old solutions (crossover) and randomly modifying a solution (mutation). The whole process is repeated until a candidate with sufficient quality is found or the maximal number of iterations has been reached. For a detailed introduction to Evolutionary Algorithms the reader is referred to, e.g. (Ashlock 2005).

Genetic Programming (GP) is a special kind of EA which uses the same principles (Koza 1992). It uses variable length tree structures to represent solution candidates and thus can be used for the automatic creation of, e.g., formulas or even computer programs or algorithms. Dispatching rules are formulas computing a priority for a job.

In the EA community, scheduling received a lot of attention (see, for example de the survey presented by Hart, Ross, and Corne (2005)), however mostly applied directly to solve instances of scheduling problems. GP has received only little attention, even though it seems very promising to use it as a hyper-heuristic (Burke et al. 2010) by generating dispatching rules, as we do in this paper. Advantages of this approach include:

1. The time-consuming GP run can be made off-line. Once a good rule was 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 shows the general sequence of actions in a GP run, which follows the standard procedure of most EAs: It starts from an initial population of randomly generated dispatching rules which are evaluated. The evaluation results (fitness) of the individuals are then used to apply the standard operators of a GP run, namely selection and creation of new dispatching rules via crossover and mutation, to form the population of the next generation. Again, the individuals of the new population

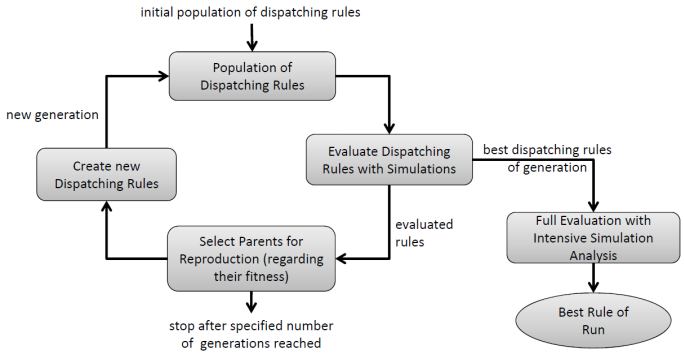

Figure 1: Genetic programming cycle to create dispatching rules

need to get evaluated and the cycle repeats until a certain maximum number of iterations (generations) have been performed.

An important design aspect for GP are the variables (terminal symbols in the tree structure) the GP is allowed to use. This is discussed later in Section 3.4.

2.1 Previous Approaches

One of the first approaches to use Genetic Programming for scheduling problems was conducted by **?**. They developed a general system to infer symbolic policy functions for distributed reactive scheduling. They used the well studied Fisher Thompson (Fisher and Thompson 1963) static job shop instances to validate their approach and their system performed very well.

In Dimopoulos and Zalzala (2001), the use of GP for the development of dispatching rules for the one-machine total tardiness problem is investigated. The generated rules outperformed standard rules like EDD and SPT, and even the Montagne rule, which was specially developed for the weighted total tardiness problem.

Geiger, Uzsoy, and Aytug (2006) also present an approach, which is capable of automatically discovering dispatching rules. They evaluated their approach in a variety of single machine environments, and discovered rules that are better than those found in the literature. In a follow-up paper Geiger and Uzsoy (2008), they used their approach to evolve dispatching rules for a single machine with batch processing.

? proposed a multiple tree adaptive heuristic for job shop scheduling, where a decision tree is used to distinguish between resources based on their load characteristics. The results of their approach exhibited better performance than existing scheduling methods.

Tay and Ho (2008) developed dispatching rules with Genetic Programming for the flexible job-shop problem, where an operation can be processed on several alternative machines. This means that not only the order but also the assignment of operations to machines has to be done. Tay and Ho try to find rules performing well for various objectives by using a linear combination of these objectives in the fitness assessment. They find rules performing better than standard rules.

In (Hildebrandt, Heger, and Scholz-Reiter 2010) GP is used to evolve dispatching rules for dynamic job shops minimizing mean flow time of jobs. Using the scenarios of Holthaus and Rajendran (Rajendran and Holthaus 1999, Holthaus and Rajendran 2000) they are able to find rules performing better than standard rules and rules manually developed by Holthaus and Rajendran.

2.2 Simulation-based Fitness Evaluation

A semiconductor manufacturing environment is a complex system, and performance of a particular dispatching rule in a stochastic and dynamic shop floor can not be judged analytically, but only via (stochastic) simulation. This implies that every dispatching rule created by GP has to be tested in a simulation scenario. This kind of problem is known as simulation-based optimization, see, e.g., (Fu 2001). One particular challenge of simulation optimization is the tremendous computing power required. In our work, we have tackled this in several ways:

1. Our framework allows the transparent utilization of multi-core processors/computers. In our experiments, we used an 8-core computer with Intel Xeon 3 GHz-CPUs.
2. All solutions of the same iteration are evaluated using the same random seed (common random numbers). To avoid a bias in optimization, different seeds are chosen for different iterations. This setting has been shown to be favourable in (**?**).
3. Because it would be too time-consuming to thoroughly evaluate each solution during optimization, we use a two-stage evaluation procedure. During optimization, each solution is evaluated only once for two simulated years. The five best solutions of each iteration (based on this rough performance estimate) are kept in memory. At the end of the run, the best found solution is identified based on additional 20 simulation runs over 6 years each, where the first year is discarded as warm-up period. This is a simplified version of the clean-up procedure suggested in (Boesel, Nelson, and Kim 2003).

2.3 System Architecture

Our framework is implemented in Java, using the ECJ library (ECJ 2009) to perform the GP-operations. To implement the manufacturing scenarios and assess the performance of dispatching rules we implemented and integrated a simple, efficient discrete-event simulation. This simulation is very roughly based on the Java-port of the job shop implementation of the SIMLIB library (Law 2007), as described in (Huffman 2001).

3 EXPERIMENTAL DESIGN

We test our GP-framework against some state-of-the-art dispatching rules on a simulation model that is derived from the publicly available MASM testbed (Feigin, Fowler, and Leachman 1996), which is described in detail next.

3.1 Scenario Description

The model is based on the dataset 4r of the MASM testbed. This set contains 36 machines in 31 work centers required to produce 7 products that have two distinguishable (reentrant) product flows. Since products sharing the same route do not differ in any other way in the dataset, it is more sensible to speak of two product categories. Regarding the work centers, some have multiple identical machines operating in parallel, other work centers possess machines with batching capabilities or that require sequence-dependent setup times.

On-time delivery is one of the major objectives within the semiconductor industry, as stated by Pfund, Mason, and Fowler (2006), which is usually measured in terms of tardiness. The tardiness of a job i is defined as $T_i = \max(C_i - D_i, 0)$, where C_i and D_i refer to a job's completion time and due date, respectively. Moreover, jobs often have different levels of importance expressed in their weight W_i. Following other related papers (see e.g., Mason et al. (2005)), the overall objective used in this paper is to minimize the average weighted tardiness $\frac{1}{n}\sum_{i=1}^{n} W_i T_i$. In this formula the summation is over all n jobs finished. The weight of a job can also be seen as a tardiness cost for delaying a job by one time unit (Vepsalainen and Morton 1987).

A few modifications are made to the original model. Operators are not considered as a restricting resource and machines are assumed to be constantly available, i.e, they are not subject to failures. Another important alteration concerns the job arrival pattern. An analysis of expected machine utilization levels given the original product mix and arrival rates revealed the presence of one clear bottleneck with an estimated utilization of only 80.7 %. To create a more challenging scenario, we use a different job arrival pattern (4.5 lots/day of product A, 10.5 lots/day of product B, a 30 %/70 % product mix) that leads to a high utilization of several work centers with diverse properties. Table 5(b) provides a description of these potential bottlenecks. Explicit attention has to be paid to the two work centers involving setups and the formation of batches as their capacity depends on the jobs available for scheduling and the scheduling strategy used. If a time-consuming setup becomes necessary or a full batch can not be formed, some of the respective work center's theoretical capacity is wasted which is why actual utilization levels are likely to exceed the lower bounds given in Table 5(b).

Job arrivals of both products follow a Poisson process with the rates given above. For an incoming job i, its weight W_i is sampled from a discrete uniform distribution on the interval $[1, 10]$ and its due date D_i is set with the so-called total work content-method, i.e. proportional to its processing time

($D_i = R_i + a * P_i$, where R_i and P_i is the release date and total processing time of job i, respectively). Allowance factors a are generated from a uniform distribution. Preliminary simulation experiments have been conducted with the FIFO priority rule (see Section 3.2) to determine the upper and lower bounds of this distribution in the attempt to reach an intermediate level of due date tightness under which most jobs can finish in time. With allowance factors ranging from 2 to 5 the application of Priority FIFO in conjunction with Setup Avoidance and Larger Batches First (see Section 3.3) has led to 34 % tardy jobs, which is deemed an adequate value. Thus, a continuous uniform distribution on the interval $[2,5]$ is used for allowance factors.

In order to evaluate our GP-based rule generation approach, resulting rules are compared to some well-known rules from the literature which are presented in the following section.

3.2 Benchmark Dispatching Rules

Many dispatching rules have been developed over the years addressing different performance measures under varying shop conditions, see for instance the works of Panwalkar and Iskander (1977), Blackstone, Phillips, and Hogg (1982), and Haupt (1989) for extensive surveys. Here, only those rules that can be expected to perform well in minimizing weighted tardiness based on results from prior studies serve as comparison rules.

3.2.1 Priority-based rules

Priority-based rules account for job weights by giving highest priority to the job with the highest weight. Thus, they are well-suited for scheduling problems involving weighted objective functions and are also widely utilized in the semiconductor industry (Pfund, Mason, and Fowler 2006). Ties between jobs of similar weight can be broken by any common dispatching rule. The following priority-based rules are included in the experiments.

Priority FIFO: Between jobs having identical weights, Priority FIFO selects the job that has been waiting for the longest time in the queue to be processed by the regarded machine. Though FIFO generally exhibits a modest tardiness performance, it is easy to implement and often serves as a benchmark in simulation studies.

Priority SPT: Priority SPT breaks ties by choosing the job with the shortest processing time for its imminent operation. Although this rule primarily aims to reduce the flow time of jobs (the difference between its completion and release time), SPT has shown to effectively minimize total tardiness when most jobs can not be meet their due dates because of a tight due date setting and/or a high shop utilization (Rajendran and Holthaus 1999).

Priority EDD: EDD resolves ties among equally weighted jobs by prioritizing the job with the earliest due date, therefore tending to decrease the maximum tardiness of all jobs. Contrary to SPT, the EDD rule is known to perform well for total tardiness when the shop is not congested and most jobs can be completed on-time (Rajendran and Holthaus 1999).

Priority ODD: Instead of the job's due date, ODD rule uses the due date of the imminent operation to prioritize jobs. ODD is generally effective in optimizing for tardiness-related criteria (Jayamohan and Rajendran 2000). Operation due dates are usually set in relation to their processing time, and the due date of the last operation equals a job's due date. Correspondingly, Priority ODD selects the job with the earliest due date of its imminent operation to break ties between jobs having the same weight.

Priority CR: The CR (critical ratio) rule schedules jobs in non-decreasing order of their critical ratio, which is calculated as follows

$$CR_i = \frac{(D_i - \tau)}{P_{\text{rem},i}}$$

where τ is the current time, and D_i and $P_{\text{rem},i}$ indicate the due date, respectively the total processing time of all remaining operations of job i. The CR rule is well-suited for minimizing jobs' tardiness as it tends to prioritize job that have less time available to finish before their due date, but also accounts for the need to schedule shorter jobs first as soon as they become tardy. Priority CR is thus often employed at semiconductor manufacturers that focus on on-time delivery (Pfund, Mason, and Fowler 2006).

3.2.2 Weighted variants of standard rules

Another popular approach to modify traditional dispatching rules in order to cope with weighted objectives is by incorporating the job weight as a factor in the calculation of priority indices (Kanet and Li 2004), and some of the more promising members of this category are included in the comparison.

WSPT The WSPT rule extends the concept of SPT in that a job's weight is divided by its imminent operation processing time for the determination of the job priorities. The processing time is thus set in relation to the job's weight and WSPT does not necessarily select the job with the shortest imminent operation if its weight is low. WSPT is selected as a comparison rules for the same reasons as Priority SPT.

WMDD WMDD is a weighted version of the MDD (modified due date) rule that computes priority values with the following formula

$$\mathrm{WMDD}_i = \frac{1}{W_i}\max(P_{\mathrm{rem},i}, D_i - \tau)$$

Jobs with a lower weighted modified due date receive higher priorities. The rule has been designed by (Kanet and Li 2004) who conclude that it is highly effective in minimizing weighted tardiness performance compared to other weighted variants of standard dispatching rules.

WMOD In consideration of the effectiveness of WMDD, it appears to be sensible to modify the MOD (modified operation due dates) rule accordingly. The resulting priority function is expressed as

$$\mathrm{WMOD}_i = \frac{1}{W_i}\max(p_{i,\mathrm{imt}}, d_{i,\mathrm{imt}} - \tau)$$

where $p_{i,\mathrm{imt}}$ and $d_{i,\mathrm{imt}}$ denote processing time and due date of the imminent operation of job i.

None of the rules mentioned so far account for sequence-dependent setup times. However, as discussed in Section 3.1 the inefficient use of machines that require setups are likely to have a huge negative effect on the overall shop performance in our scenario. To alleviate this problem, **Setup Avoidance** is employed in conjunction with all of the above dispatching rules, which means that jobs which do not require any setup always receive highest priority.

3.2.3 ATCS (Apparent Tardiness Cost with Setups)

The ATCS rule has been developed by Lee, Bhaskaran, and Pinedo (1997) as an extension to the ATC rule by Vepsalainen and Morton (1987) to explicitly considers sequence-dependent setup times in the prioritization of jobs. ATC has been designed specifically with the aim to minimize weighted tardiness, and since setups are critical in the scenario used in this study, ATCS appears to be a highly competitive rule. The priority index ATCS_i is calculated as

$$\mathrm{ATCS}_i = \frac{W_i}{p_{i,\mathrm{imt}}}\exp\left(-\frac{\max(d_{i,\mathrm{imt}} - p_{i,\mathrm{imt}} - \tau, 0)}{k_1\overline{p}_{\mathrm{imt}}}\right)\exp\left(-\frac{s_{\mathrm{cur},i}}{k_2\overline{s}}\right)$$

The job with the highest index is selected for processing by this rule. In the formula, $s_{\mathrm{cur},i}$ refers to the (sequence-dependent) time required to set up the machine from the current state to the state required for processing job i, $\overline{p}_{\mathrm{imt}}$ and $\overline{s}$ indicate the average operation processing time respectively setup time of all jobs awaiting service, and other variables have the previously introduced meaning. k_1 and k_2 are scaling parameters, for which appropriate values have to be determined. To illustrate, ATCS reverts to the operation of ATC when k_2 is set to a very high value. Similarly, ATCS converges to the WSPT rule if high values are assigned to both parameters.

3.3 Batching Policies

To control batch formation we use two heuristics. The first is greedy batching with a certain minimum batch size (MBS), and the second is larger batches first (LBF).

Table 1: GP parameters used

Name	Value
population size	1000
generations	50
crossover proportion	90 %
reproduction proportion	10 %
selection method	tournament selection (size 7)
creation type	ramped half-and-half (min depth 2, max depth 6)
max. depth for crossover	17
function set	$+$, $-$, $\times$, $\div$, max, if3
basic terminal set	PT, RemProcTime, OpProcAvg, Weight, TimeInQueue, TimeTillDue, Slack 0, 1
setup-related terminals	SetupTime, SetupAvg
batching-related terminals	NumJobsSameBatchFamily

MBS Batch formation uses a greedy procedure and works by first sequencing 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 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(n) where n is the minimum batch size. If $n > 1$ we filter the list of waiting jobs to include only those jobs belonging to a batch family with at least n jobs waiting.

Larger Batches First The LBF procedure forms a batch as large as possible for each batch family. Then the largest of these batches is started. Should there be a tie, it is resolved by following the just described greedy procedure selecting the batch family of the job having highest priority.

3.4 Rule Components and GP Parameters

In order to keep results between our benchmark rules and the results of GP comparable we chose the set of terminals to only include information also used in the benchmark rules. Our terminal set therefore consists of the 10 terms listed below plus 0 and 1 as constants. As a dispatching rule gets evaluated each time a machine becomes idle to choose one of its currently waiting jobs, all terms are relative to the job j, which is to be evaluated and assigned a priority.

- **constants**: as constant values we use $\{0, 1\}$
- **PT**: Processing Time of j's current operation
- **RemProcTime**: sum of processing times of all operations left for j
- **OpProcAvg**: average processing time of all waiting jobs
- **Weight**: job j's weight
- **TimeInQueue**: time j spent in current queue in front of machine m: $\text{TimeInQueue}_{j,m} = \tau - r_{j,m}$, where $r_{j,m}$ is the point in time j entered the queue
- **TimeTillDue**: $\text{TimeTillDue}_j = D_j - \tau$
- **Slack**: the slack denotes the difference between the time a job gets due and its remaining processing time to finish it $sl_j = \text{TimeTillDue}_j - \text{RemProcTime}_j$. If the slack becomes negative a job can not possibly meet its due date D_j.
- **SetupTime**: setup time if job j would be started next; this terminal is always 0 for all machines without sequence-dependent setups
- **SetupAvg**: average setup time of all waiting jobs
- **NumJobsSameBatchFamily**: number of jobs in q that are compatible with the batch family required by j's current operation; for all non-batch-machines this terminal always produces 1

Our function set consists of the four basic arithmetic operations augmented by the maximum function and a ternary version of if-then-else (if $a >= 0$ then b else c; a, b, c are sub-expressions evolved with GP).

All GP parameters used are summarized in Table 1. Besides choosing the set of terminals and non-terminals, we did not make any serious attempts to optimize these parameters. The parameter

Table 2: Results of benchmark rules. All times in the table are in minutes, numbers in brackets indicate 95 % confidence intervals.

seq. rule	batching	flowMean	tardMean	tardPercentage	wTardMean
PR-FIFO	LBF	12296 (± 98)	3859 (± 82)	33.2% (±0.2%)	**6044 (±121)**
PR-SPT	LBF	12350 (±109)	3896 (± 91)	33.9% (±0.3%)	**6161 (±140)**
PR-EDD	LBF	12278 (± 93)	3785 (± 75)	32.8% (±0.2%)	**5842 (±119)**
PR-ODD	LBF	12291 (±101)	3792 (± 83)	33.0% (±0.3%)	**5841 (±128)**
PR-CR	LBF	12283 (± 99)	3781 (± 81)	33.1% (±0.2%)	**5810 (±120)**
WSPT	LBF	12383 (±108)	3910 (± 88)	34.1% (±0.3%)	**6217 (±139)**
WMDD	MBS(5)	11485 (±110)	1766 (± 91)	33.6% (±0.4%)	**4210 (±215)**
WMOD	MBS(5)	11089 (±122)	1054 (± 94)	42.3% (±1.2%)	**2557 (±177)**
ATCS(k1=4.5;k2=0.01)	MBS(5)	10963 (±123)	1003 (± 93)	34.7% (±1.1%)	**2336 (±168)**

settings are mostly ECJs defaults following recommendations from (Koza 1992). We found these settings to work well in our experiments.

4 EXPERIMENTAL RESULTS

4.1 Benchmark rules

We first made a conventional simulation study to find out the best system configuration using standard rules. We tested all combinations of dispatching rules (PR-FIFO, PR-SPT, PR-EDD, PR-ODD, PR-CR, WSPT, WMDD, WMOD) and batching policies (MBS(1), MBS(2), ..., MBS(8), LBF), totaling $8 \times 9 = 72$ system configurations. To find the best configuration we used the same procedure also used for the "full" evaluations of GP: 20 independent replications of a simulation time of 6 years each (ignoring data from jobs finished in the first year). As stated before each of these experiments used setup avoidance.

To find the best parameter settings for ATCS we performed experiments using 24 different values for k_1 (0.0001, 0.001, 0.01, 0.1, 0.5, 1.0, 1.5, ..., 6.5, 7, 8, 9, 10, 20, 50, 100) and 8 different values for k_2 (0.0001, 0.001, 0.01, 0.1, 0.25, 0.5, 1.0, 1.5). We systematically tested each combination of k_1 and k_2 with each of the 9 batching variants, leading to $24 \times 8 \times 9 = 1728$ different system configurations.

Results of these experiments are listed in Table 2. The table lists for each sequencing rule the best-performing combination with a batching heuristic regarding weighted tardiness (last column). The table also shows some other performance measures of interest.

4.2 GP

We performed 10 independent GP runs. In each of the runs we saved the 5 best individuals of each generation. We therefore end up with a set of $10 \times 50 \times 5 = 2500$ dispatching rules, which are fully evaluated with 20 replications of 6 years of simulation time (ignoring data from the first year) to select the best rule. As a result of this we identified GPRuleSize199 as the best-performing rule.

This rule is very long and hard to analyze. To also find shorter and therefore easier rules yet performing not much worse, we identified the set of Pareto-optimal rules regarding rule size and mean weighted tardiness out of these 2500 rules. Doing so gives a set of just 16 Pareto-optimal rules, i.e. each of these rules achieves the best performance for its length, longer rules are only accepted it they yield better weighted tardiness results. Out of these 16 rules 12, starting with a rule of size 9, perform better than the best benchmark rule ATCS. These rules and their performance are listed in Table 3, the formula of selected rules are given in the appendix.

As can be seen in this table GPRuleSize199 improves mean weighted tardiness over ATCS by 70% and produces very good results for the other objectives as well. This is shown in more detail in Table 4. Here we compare the results of the best GP rule with the best benchmark rule for *each* objective. GPRuleSize199 is not just superior in the objective used for fitness evaluation in the course of a GP run: weighted tardiness (improvement of 70%). Also for "mean tardiness" (improvement of 59%) and "percentage of tardy jobs" (improvement of 9 percentage points) we get clearly superior results. Even the mean flowtime results are not worse than the best benchmark rule for this objective. Therefore in our test scenario GP was able to find a rule dominating our benchmark rules regarding

Table 3: Results of best GP rules; all times are again in minutes; numbers in brackets are 95 % confidence intervals. These rules are always used with MBS(1).

seq. rule	flowMean	tardMean	tardPercentage	wTardMean
GPRuleSize09	10026 (± 97)	572 (± 55)	26.5% (±1.1%)	**1669 (±142)**
GPRuleSize12	10090 (±104)	872 (± 65)	21.6% (±0.7%)	**1458 (±101)**
GPRuleSize16	10025 (± 98)	375 (± 60)	22.3% (±1.6%)	**1018 (±133)**
GPRuleSize20	10077 (±107)	405 (± 66)	19.1% (±1.5%)	**912 (±128)**
GPRuleSize33	10171 (± 88)	368 (± 53)	24.0% (±1.6%)	**807 (±101)**
GPRuleSize43	10142 (± 95)	347 (± 58)	24.1% (±1.8%)	**766 (±110)**
GPRuleSize98	10252 (±108)	383 (± 65)	21.3% (±2.1%)	**782 (±129)**
GPRuleSize99	10097 (± 96)	373 (± 58)	20.1% (±1.4%)	**728 (±106)**
GPRuleSize110	10173 (± 88)	348 (± 53)	23.9% (±1.7%)	**735 (±101)**
GPRuleSize120	10117 (±101)	398 (± 62)	18.1% (±1.6%)	**721 (±108)**
GPRuleSize122	10152 (± 96)	360 (± 58)	18.9% (±1.7%)	**715 (±108)**
GPRuleSize199	10105 (± 95)	397 (± 57)	19.4% (±1.3%)	**696 (± 95)**

the objectives just mentioned: GPRuleSize199 can be used to drastically improve due-date related objectives without an increase in mean flowtime.

Table 4: Results of best GP rule and best benchmark rule for *each* objective (note that the parameters for ATCS are not the same across different objectives).

	flowMean	tardMean	tardPercentage	wTardMean
best benchmark	10963 (±123)	975 (± 94)	28.1% (± 0.3%)	2336 (±168)
rule	ATCS;MBS(5)	ATCS;MBS(5)	ATCS;LBF	ATCS;MBS(5)
GPRuleSize199	10105 (± 95)	397 (± 57)	19.4% (±1.3%)	696 (± 95)

5 ROBUSTNESS OF GP RULES

An interesting question is, whether these results are robust to changing conditions. Therefore we conducted experiments varying the utilization level and the product mix. Besides the high utilization of 93.8 % we used in the experiments before we also use a more moderate load level of 85 %. We also change the product mix from 30/70 to 70/30. Altogether we get the scenarios shown in Table 5(a). For each of the four scenarios the table shows the start rates of product A and product B, i.e. the setting used in the GP run uses 4.5 lot starts per day of product A and 10.5 lot starts per day of product B.

Table 5: Experimental settings for the robustness analysis.

(a) start rates

		product mix	
		30 / 70	**70 / 30**
util.	**85%**	4.08 / 9.52	4.29 / 1.84
	93.80%	4.5 / 10.5	4.74 / 2.03
max lots /day:		16	7.22

(b) bottlenecks with product mix 30/70

Machine	Util.
AME135	93.8%
D1-9 (3 machines in group)	91.1%
DFC4	89.1%
PE1-5 (3 machines in group; setups)	74.3%
LPS1	71.9%
QLESS (batch)	64.8%

(c) bottlenecks with product mix 70/30

Machine	Util.
DFC4	93.8%
D1-9 (3 machines in group)	69.2%
ASM2	65.8%
WET3	56.0%
PE1-5 (3 machines in group; setups)	49.4%
QLESS (batch)	47.5%

Changing the product mix also changes the location of bottlenecks on the shop floor. Therefore Tables 5(b) and 5(c) list the six most critical machine groups for a product mix of 30/70 and 70/30 respectively. As can be seen the first mix yields more machines to be at a very high utilization, whereas for the second product mix there is a single bottleneck and remaining machines are only at a moderate load level.

For each of the 3 scenarios not used in the GP run we identified the best combination of sequencing rule and batching heuristic as described in Section 4.1. However we did not try to find the best parameter

Table 6: Results of robustness experiments. Table cells with bold numbers indicate results better than the benchmark rule (lower mean and non-overlapping confidence intervals).

	85%, 30/70	93.8%, 30/70	85%, 70/30	93.8%, 70/30
best benchmark rules	652 (± 48) ATCS;MBS(4)	2336 (±168) ATCS;MBS(5)	216 (± 30) WMOD;MBS(1)	1245 (±140) WMOD;MBS(3)
GPRuleSize09	**451 (± 17)**	**1669 (±142)**	644 (± 19)	**868 (± 61)**
GPRuleSize12	**483 (± 17)**	**1458 (±101)**	358 (± 14)	**876 (± 78)**
GPRuleSize16	**187 (± 11)**	**1018 (±133)**	387 (± 10)	**718 (±102)**
GPRuleSize20	**144 (± 8)**	**912 (±128)**	225 (± 8)	**552 (± 87)**
GPRuleSize33	**127 (± 8)**	**807 (±101)**	**84 (± 3)**	**273 (± 60)**
GPRuleSize43	**96 (± 7)**	**766 (±110)**	**71 (± 3)**	**244 (± 57)**
GPRuleSize98	**47 (± 6)**	**782 (±129)**	**51 (± 3)**	**206 (± 57)**
GPRuleSize99	**81 (± 7)**	**728 (±106)**	**135 (± 4)**	**345 (± 66)**
GPRuleSize110	**91 (± 6)**	**735 (±101)**	**68 (± 3)**	**240 (± 51)**
GPRuleSize120	**83 (± 6)**	**721 (±108)**	**98 (± 5)**	**369 (± 82)**
GPRuleSize122	**62 (± 7)**	**715 (±108)**	**73 (± 3)**	**240 (± 58)**
GPRuleSize199	**95 (± 7)**	**696 (± 95)**	**98 (± 4)**	**279 (± 59)**

values for ATCS, but instead used the best parameters identified before and treat ATCS(k_1=4.5,k_2=0.01) as any other standard rule. We therefore made $9 \times 9 = 81$ experiments for each of the 3 scenarios.

Results of these experiments are shown in Table 6 listing mean tardiness performance of the best GP rules and the best benchmark rule of each scenario. The third column, 93.8 % utilization, product mix 30/70, is the scenario of the GP run, therefore its results are identical to those already presented in Tables 3 and 2.

For the conventional rules the good performance of the WMOD rules is remarkable. This non-parameterized rule beats ATCS in some scenarios not used to optimize ATCS' parameters. Tuning the parameters for each scenario would probably change the situation in favour of ATCS again, but is computationally intensive.

GP rules show very good performance irrespectively of the scenario. Rules of size 33 and above exhibit good generalization abilities. Based on the results of our experiments GPRuleSize98 is the most robust, as it reaches the lowest mean weighted tardiness in the three altered scenarios and is not significantly worse than GPRuleSize199 in our original scenario.

6 CONCLUSION

Semiconductor manufacturing is a complex dynamic and stochastic process, usually involving re-entrant flows, parallel machines, sequence-dependent setup times and batching. In this paper, we have applied simulation-based Genetic Programming (GP) to automatically generate dispatching rules for scheduling in semiconductor manufacturing. As such, it is the first paper to successfully and automatically generate dispatching rules for such complex production environments.

Based on an example with 36 machines and two product categories, we compare the dispatching rules produced by GP with a large number of state-of-the-art dispatching rules from the literature. The result is impressive: Compared to the best performing benchmark rule, ATCS, the GP generated rule is able to reduce the mean weighted tardiness by 70%. Furthermore, the rule performs robustly and significantly better than all other benchmark rules even when the product mix and/or utilization level are changed.

There are many avenues for future research. First, it would be interesting to see whether even better rules could be generated for other objective functions, if they are specifically evolved for this purpose. Second, the strategies implemented to reduce the computational effort spent on simulation could be refined, integrating, e.g., techniques from ranking and selection. Third, the set of variables (terminal symbols) available to GP could be expanded, or different dispatching rules could be allowed for different machines. Finally, simulation-based GP could run continuously during production in real-time, constantly adapting the dispatching rule to a changing environment.

ACKNOWLEDGMENTS

The authors would like to gratefully acknowledge funding from the German Research Council (DFG) under grant BR 1592/7-1 and SCHO 540/17-1.

A APPENDIX

Selected rules found by GP are listed below. In these formulas all terminals are abbreviated as single letter variables. The variables stand for the following terminals: $p =$ PT, $P =$ OpProcAvg, $w =$ Weight, $r =$ RemProcTime, $q =$ TimeInQueue, $d =$ TimeTillDue, $L =$ Slack, $s =$ SetupTime, $S =$ SetupAvg, $b =$ NumJobsSameBatchFamily.

Name	Rule
GPRuleSize09	$w/\max(L,P)-s+b$
GPRuleSize33	$w+b-\max(1,s,w+d/(\max(1,s)*r)-b-\max(1,s,r*(w-b)/d))-1$
GPRuleSize98	$(\max(1,r)-\max(1,r,L),w,b)*b*\max(r/L+\max(-(b-L,w,b)+s+b,S+b*(\max(1,r)-\max(L,d),w,b)-s-\max(1,r,L)+\max(1,r)+1)*(b-L,w,b)-s,S+b*(\max(1,r)-L,w,b)*(2*r/L-s)+r/L-s+1)$

REFERENCES

Ashlock, D. 2005. *Evolutionary computation for modeling and optimization*. Springer.

Baker, K. R. 1977. An experimental study of the effectiveness of rolling schedules in production planning. *Decision Sciences* 8 (1): 19–27.

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.

Boesel, J., B. L. Nelson, and S.-H. Kim. 2003. Using ranking and selection to "clean up" after simulation optimization. *Operations Research* 51 (5): 814–825.

Branke, J., and D. C. Mattfeld. 2005. Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research* 43 (15): 3103–3129.

Burke, E. K., M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward. 2010. A classification of hyper-heuristics approaches. In *Handbook of Meta-heuristics* (2nd ed.)., ed. M. Gendreau and J.-Y. Potvin. Springer. Forthcoming.

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 (6): 489–498.

ECJ 2009. A Java-based Evolutionary Computation Research System, project homepage. http://cs.gmu.edu/~eclab/projects/ecj/, last accessed 15th January 2010.

Feigin, G., J. Fowler, and R. Leachman. 1996. Masm test data sets. http://www.eas.asu.edu/~masmlab. Last accessed on 28th May 2009.

Fisher, H., and G. L. Thompson. 1963. Probabilistic learning combinations of local job-shop scheduling rules. In *Industrial Scheduling*, ed. J. F. Muth and G. L. Thompson, 225–251. Prentice-Hall.

Fu, M. 2001. Simulation optimization. In *Winter Simulation Conference*, 53–61: IEEE.

Geiger, C., R. Uzsoy, and H. Aytug. 2006. Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach. *Journal of Scheduling* 9 (1): 7–34.

Geiger, C. D., and R. Uzsoy. 2008. Learning effective dispatching rules for batch processor scheduling. *International Journal of Production Research* 46 (6): 1431–1454.

Hart, E., P. Ross, and D. Corne. 2005. Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines* 6 (2): 191–220.

Haupt, R. 1989. A survey of priority rule-based scheduling. *OR Spektrum* 11 (1): 3–16.

Hildebrandt, T., J. Heger, and B. Scholz-Reiter. 2010. Towards improved dispatching rules for complex shop floor scenarios—a genetic programming approach. In *Genetic and Evolutionary Computation Conference*. (accepted paper, to appear).

Holthaus, O., and C. Rajendran. 2000. Efficient jobshop dispatching rules: further developments. *Production Planning & Control* 11 (2): 171–178.

Huffman, B. J. 2001. An object-oriented version of SIMLIB (a simple simulation package). *INFORMS Transactions on Education* 2 (1): 1–15.

Jayamohan, M. S., and C. Rajendran. 2000. New dispatching rules for shop scheduling: a step forward. *International Journal of Production Research* 38 (3): 563–586.

Kanet, J. J., and X. Li. 2004. A weighted modified due date rule for sequencing to minimize weighted tardiness. *Journal of Scheduling* 7:261–276.

Koza, J. R. 1992. *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press.

Law, A. M. 2007. *Simulation modeling and analysis*. 4. ed. Boston, USA et al.: McGraw-Hill.

Lee, Y. H., K. Bhaskaran, and M. Pinedo. 1997. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions* 29 (1): 45–52.

Mason, S. J., J. W. Fowler, W. M. Carlyle, and D. C. Montgomery. 2005. Heuristics for minimizing total weighted tardiness in complex job shops. *International Journal of Production Research* 43 (10): 1943–1963.

Panwalkar, S. S., and W. Iskander. 1977. A survey of scheduling rules. *Operations Research* 25 (1): 45–61.

Pfund, M. E., S. J. Mason, and J. W. Fowler. 2006. *Handbook of production scheduling*, Volume 89, Chapter Semiconductor manufacturing scheduling and dispatching, 213–241. Springer US.

Rajendran, C., and O. Holthaus. 1999, July. A comparative study of dispatching rules in dynamic flowshops and jobshops. *European Journal of Operational Research* 116 (1): 156–170.

Tay, J. C., and N. B. Ho. 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering* 54 (3): 453–473.

Uzsoy, R., C. Y. Lee, and L. A. Martin-Vega. 1992. A review of production planning and scheduling models in the semiconductor industry part i: system characteristics, performance evaluation, and production planning. *IIE Transactions* 24:47–60.

Vepsalainen, A. P. J., and T. E. Morton. 1987, Aug., 1987. Priority rules for job shops with weighted tardiness costs. *Management Science* 33 (8): 1035–1047.

AUTHOR BIOGRAPHIES

JÜRGEN BRANKE has a chair for Operational Research and Systems at the Warwick Business School, UK. Before this, he was teaching at the University of Karlsruhe, Germany and worked as a senior scientist at Icosystem Inc., France. He is an associate editor for the Evolutionary Computation Journal and for the Journal of Heuristics. His research interests are nature inspired optimisation; logistics and scheduling; organic computing; multiobjective optimisation and decision making; optimisation in presence of uncertainty and simulation-based optimisation. His email address is <juergen.branke@wbs.ac.uk>.

CHRISTOPH PICKARDT received the MEngSt degree from the University of Auckland, New Zealand and Dipl.-Wi.-Ing. degree from the University of Karlsruhe, Germany. He is a PhD candidate at the Warwick Business School, UK. His email address is <christoph.pickardt@warwick.ac.uk>.

TORSTEN HILDEBRANDT works as a research scientist at the Bremen Institute of Production and Logistics at the University of Bremen with Prof. Scholz-Reiter. His email address is <hil@biba.uni-bremen.de>.

JENS HEGER graduated from the University of Paderborn in 2006 with a M.Sc. degree in computer science. Since 01/2007 he has been working as a research scientist at the University of Bremen with Prof. Scholz-Reiter. His email address is <heg@biba.uni-bremen.de>.

BERND SCHOLZ-REITER holds the chair of Planning and Control of Production Systems in the Department of Manufacturing Engineering at the University of Bremen. He is director of the Bremer Institut für Produktion and Logistik (BIBA), where he works in applied and industrial contract research. Prof. Scholz-Reiter is the speaker of the Collaborative Research Centre 637 ”Autonomous Cooperating Logistic Processes - A Paradigm Shift and its Limitations”, speaker of the International Graduate School for Dynamics in Logistics at the University of Bremen and speaker of the Bremen Research Cluster for Dynamics in Logistics. His email address is <bsr@biba.uni-bremen.de>.