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

LEARNING DISPATCHING RULES FOR ENERGY-AWARE SCHEDULING OF JOBS ON A SINGLE BATCH PROCESSING MACHINE

Daniel Sascha Schorn
Lars Mönch

Department of Mathematics and Computer Science
University of Hagen
Universitätsstraße 1
Hagen, 58097, GERMANY

ABSTRACT

In this paper, we consider a scheduling problem for a single batch processing machine in semiconductor wafer fabrication facilities (wafer fabs). An integrated objective function that combines the total weighted tardiness (TWT) and the electricity cost (EC) is considered. A time-of-use (TOU) tariff is assumed. A genetic programming (GP) procedure is proposed to automatically discover dispatching rules for list scheduling approaches. Results of designed computational experiments demonstrate that the learned dispatching rules lead to high-quality schedules in a short amount of computing time.

1 INTRODUCTION

Integrated circuits (ICs) on silicon wafers, thin discs made from silicon or gallium arsenide, are produced in semiconductor manufacturing. Wafer fabrication, sort, assembly, and final test are the major production stages (Mönch et al. 2013). Among the four stages, the wafer fabrication stage that is performed in wafer fabs is the most challenging one. Although dispatching is the major production control approach in wafer fabs, with the recent dramatic increase in computer efficiency, scheduling approaches have become more competitive (Mönch et al. 2011). It is well-known that the semiconductor industry consumes more energy than other industries such as the steel or petrochemical industry (Yu et al. 2017; Mönch et al. 2018). Therefore, it is desirable that sustainability issues are taken into account in scheduling approaches for wafer fabs.

Up to now energy-aware scheduling approaches are only rarely discussed for wafer fabs. We are only aware of Rocholl et al. (2020) where a scheduling problem for the diffusion furnace tool group is studied. A TOU tariff is assumed in this paper. Genetic algorithms (GAs) are used to compute the set of all Pareto-optimal schedules for the TWT and EC measures. Diffusion furnaces belong to the machines with the largest energy consumption in wafer fabs (Singapore Government 2022) since the diffusion process is a high temperature process that disperses material on the wafer surface. Hence, from a sustainability point of view it is crucial to look for efficient energy-aware scheduling approaches for diffusion furnaces.

In the present paper, we consider a scheduling problem for a single batch processing machine with a TOU tariff. A GP approach is proposed to automatically discover dispatching rules that can be applied within list scheduling approaches. In contrast to Rocholl et al. (2020), we consider an integrated objective function based on the TWT and the EC measure, respectively. We will demonstrate by designed computational experiments that the learned dispatching rules are able to compute high-quality schedules for this model problem under many experimental conditions.

The remainder of this paper is organized as follows. In the next section, we will describe the problem and discuss related work. The proposed learning approach will be discussed in Section 3. This includes the

discussion of a reference approach based on the apparent tardiness cost (ATC) dispatching rule. Results of computational experiments with the automatically discovered dispatching rules are presented in Section 4. Finally, conclusions and future work directions are discussed in Section 5.

2 PROBLEM SETTING

2.1 Problem Statement

A finite scheduling horizon consisting of $t = 1, \dots, T$ periods of equal length is assumed. We consider n jobs that are processed on a single batch processing machine. A batch is a group of jobs that are processed at the same time, i.e. in parallel, on the same machine. This setting is called p-batching. Each job j has a due date d_j and a weight w_j. All the jobs are ready to be processed at the beginning of the scheduling horizon. The jobs belong to incompatible families. The families are labeled by $f = 1, \dots, F$. Only jobs of the same family f can be batched together. All jobs belonging to a family f have the same processing time p_f. We assume that n_f jobs are in family f. The family of job j is $f(j)$. The maximum batch size B is measured in number of jobs. After a batch is started, it cannot be interrupted. The EC is modeled as a piecewise constant function over the periods of the scheduling horizon, i.e., we have $EC = \sum_{t=1}^{T} e(t) z_t$ where $e(t)$ is the EC in period t and z_t is an indicator variable that is 1 if a batch is processed in period t and 0 otherwise. The TWT is given by $TWT = \sum_{j=1}^{n} w_j \max(C_j - d_j, 0)$, where C_j is the completion time of job j.

Using the three-field notation from deterministic machine scheduling theory (Graham et al. 1979), the scheduling problem at hand can be represented by

$$1|p - batch, incompatible|\lambda TWT + (1 - \lambda)EC, \quad (1)$$

where *p-batch* refers to parallel batch processing and *incompatible* to incompatible job families. The parameter $\lambda \epsilon [0,1]$ is used to balance the two objectives that are in conflict. The integrated performance measure for (1) is not regular since including idle time in a schedule might be beneficial for the EC measure.

It is easy to see that each optimal solution of (1) is Pareto-optimal too. Since the $1||TWT$ problem, a special case of the $1|p - batch, incompatible|TWT$ problem for $B = 1$, is already NP-hard (Lawler 1977) the bi-criteria scheduling problem (1) is also NP-hard (T'kindt and Billaut 2002). Hence, we have to look for efficient heuristics.

2.2 Related Work

We refer to Fowler and Mönch (2022) for a recent survey of scheduling methods for p-batching problems. A GP procedure (Nguyen et al. 2017) to learn dispatching rules for scheduling problems for a single batch processing machine with incompatible families and the total completion time (TC) and total tardiness (TT) measure is proposed by Geiger and Uzsoy (2008). Their approach based on the Scheduling Rule Discovery and Parallel Learning System (SCRUPLES) from Geiger et al. (2006) outperforms the best performing algorithms from Uzsoy (1995) and is competitive with the Batched ATC (BATC) rule of Mehta and Uzsoy (1998). However, only manufacturing-related performance measures are considered by Geiger and Uzsoy (2008). In the present paper, we apply GP to a scheduling problem with an energy-aware, non-regular performance measure. Hildebrandt et al. (2014) and Kück et al. (2017) use GP to discover appropriate dispatching rules for a wafer fab. Discrete-event simulation is used to assess the quality of the dispatching rules. However, sustainability issues are not taken into account in these papers.

There are only very few papers that tackle energy-aware scheduling problems for batch processing machines with the TWT performance measure. Liu (2014) consider a bi-criteria scheduling problem for a single batch processing machine. TWT and an EC-related measure are considered. The set of all Pareto-optimal schedules is computed. However, the problem does not contain incompatible families and a TOU tariff. Moreover, several hybrid GAs are proposed by Rocholl et al. (2020) to tackle a parallel-machine version of problem (1). Again, the set of Pareto-optimal schedules is computed.

In the present paper, we consider a special case of the problem of Rocholl et al. (2020). However, only an integrated performance measure is used. We are not interested in computing the set of all Pareto-optimal schedules. The main goal is to automatically determine appropriate dispatching rules for list scheduling approaches.

3 LEARNING APPROACH FOR ENERGY-AWARE DISPATCHING RULES

3.1 List Scheduling Approach

We start by sketching a list scheduling scheme for scheduling problem (1). A dispatching rule is the main ingredient of a list scheduling approach. The jobs are sorted with respect to the index value of the rule. The job with the highest index value is selected to be processed next on an available machine. Hence, appropriate problem-specific dispatching rules are required. We use the BATC rule proposed by Mehta and Uzsoy (1998) to form and sequence batches which is known to provide often high-quality schedules for the TWT measure (cf., for instance, Almeder and Mönch 2011). The BATC rule is based on the ATC index:

$$I_j(t) := w_j/p_{f(j)} e^{\left(-\max\left(d_j - p_{f(j)} - t, 0\right)/\kappa \bar{p}\right)}, \tag{2}$$

where t is the point in time where the dispatch decision is made, κ is a look-ahead parameter, and $\bar{p}$ is the average processing time of the unscheduled jobs.

The jobs of each family are sorted with the respect to (2) in non-increasing order. The number of unscheduled jobs of family f is denoted by $l(f)$. The first $\min(l(f), B)$ jobs of family f are used to form a batch $b(f)$. We then compute the BATC index:

$$I_{BATC}(f,t) = \sum_{j \epsilon b(f)} I_j(t). \tag{3}$$

The batch of the family with the largest BATC value (3) is chosen to be processed next. We refer to this batch as b_{next}. Since it might be beneficial to include idle time into the schedule to avoid periods with large EC values, a decision theory heuristic (DTH) similar to the one proposed by Mönch et al. (2005) is applied. For a given batch that can be scheduled in period t it determines the consequences of an idle period, i.e. of postponing the batch, with respect to the TWT and EC measures. The DTH for a given batch b_{next} and an available machine at time t works as follows:

DTH Procedure

1. Compute the increase of the TWT and EC values of the schedule caused by the scheduling decision for batch b_{next}. We obtain $TWT(b_{next}, t) := \sum_{j \epsilon b_{next}} w_j \max\left(t + p_{f(j)} - d_j, 0\right)$ and $EC(b_{next}, t) := \sum_{\tau=t}^{t+p_{f(j)}} e(\tau)$ where t is the time for making the scheduling decision.
2. In addition to b_{next}, it is desirable to consider the set of unscheduled jobs. Therefore, we estimate an average delay P^* of the unscheduled jobs J_u excluding the jobs of b_{next} by $P^* = 1/B \left(\sum_{j \in J_{u-b_{next}}} p_{f(j)}\right)$. The estimated TWT increase is given by $TWT(J_u, t) := \sum_{j \in J_{u-b_{next}}} w_j \max\left(t + P^* + p_{f(j)} - d_j, 0\right)$ and the estimated EC increase is $EC(J_u, t) := \sum_{\tau=t+p_{f(j)}+P^*}^{t_{end}} e(\tau)$ where $t_{end}(t) := \min\left(T, \left\lceil t + p_{avg} + P\,|J_u|/B \right\rceil\right)$. Here, p_{avg} is the average processing time of all jobs, and P is the current number of right-shifts in periods.
3. We look at maximum $P_{\max} := \lceil p_{avg}/2 \rceil$ periods into the future. For all $P \leq P_{\max}$ we compute $TWT(b_{next}, t+P), EC(b_{next}, t+P), TWT(J_u, t+P)$, and $EC(J_u, t+P)$. Initially, we set $P := 1$. The quantities

$$TWT_{\triangle}(b_{next}, t, P) := TWT(b_{next}, t+P) - TWT(b_{next}, t)$$

$$EC_{\triangle}(b_{next}, t, P) \coloneqq EC(b_{next}, t+P) - EC(b_{next}, t)$$

$$TWT_{\triangle}(J_u, t, P) \coloneqq TWT(J_u, t+P) - TWT(J_u, t)$$

$$EC_{\triangle}(J_u, t, P) \coloneqq EC(J_u, t+P) - EC(J_u, t)$$

are computed. Next, we introduce the increase of the objective function (OF) given by

$$OF_{\triangle}(b_{next}, t, P) \coloneqq \lambda TWT_{\triangle}(b_{next}, t, P) + (1-\lambda)\, EC_{\triangle}(b_{next}, t, P)$$

$$OF_{\triangle}(J_u, t, P) \coloneqq \lambda TWT_{\triangle}(J_u, t, P) + (1-\lambda)\, EC_{\triangle}(J_u, t, P).$$

We introduce the idle period when $OF_{\triangle}(b_{next}, t, P) + OF_{\triangle}(J_u, t, P) < 0$ holds, i.e., when introducing idle time for P periods is beneficial with respect to the integrated objective function. The approach goes to Step 4 when we decide for a given t and $P \leq P_{\max}$ to include P idle periods into the schedule. If such a P value does not exist we schedule b_{next} at time t and terminate the algorithm.

4. Repeat the Steps 1.-3. for $t \coloneqq t + P$ until t does not exceed the scheduling horizon.

Note that the DTH scheme works for an arbitrary index (2) and its batched version (3). Hence, it will be applied in the GP approach too. When the DTH is applied with the BATC rule, we refer to it as BATC-DTH scheme.

3.2 GP Approach

GP is an approach to construct tree structures based on a given set of primitives (Michalewicz 1996; Nguyen et al. 2017). These primitives belong to a set of relational and conditional functions denoted by F and a set of problem-specific terminals T. The relational and conditional functions are unary and binary operators and functions. Variables and numerical constants form the set of terminals. Terminals model the attribute values within the priority indices of dispatching rules.

The logical expression of a priority index can be transformed into an intermediate representation using prefix notation. The prefix notation is based on the idea that the function is written before the arguments it operates on. On the one hand, an expression tree can be derived from the priority index in prefix notation in an automated manner. Functions correspond to nodes in the tree whereas terminals form the leaves of the expression tree. On the other hand, an expression tree can be transformed automatically into an expression in prefix notation by carrying out an preorder traversal of the tree. As a result, a priority index can be represented by an expression tree.

A GP is a special GA type that operates on tree structures of variable lengths to represent solution candidates. A GP starts from a candidate set of solutions, a so-called population. The individuals belonging to the population are called chromosomes. Each chromosome represents a dispatching rule. The chromosomes of the initial population are randomly chosen. Genetic operators such as crossover and mutation are applied. Two chromosomes are randomly chosen with a crossover probability pc. A subtree is randomly determined in the tree that belongs to a parent chromosome. The two subtrees are swapped between the two parent chromosomes. The mutation operator first randomly chooses two disjoint subtrees from the tree that is associated with the chromosome with a probability of occurrence of pm. The selected subtrees are then swapped. Note that the chosen crossover and mutation operators ensure feasibility of the expression trees. Crossover and mutation are illustrated in Figure 1. A steady-state replacement with a replacement rate of rr is used. Roulette wheel selection is applied.

The applied terminals for scheduling problem (1) are summarized in Table 1. We see that both TWT- and EC-related terminals are used. The set of functions is shown in Table 2. In order to assess the fitness

of a chromosome, the chromosome is decoded, i.e., the priority index is computed. Based on this index and its batched counterpart, an instance of (1) is solved by applying list scheduling together with the DTH procedure sketched in Subsection 3.1. Note that we have to replace the objective function $\lambda TWT + (1-\lambda)EC$ by $\lambda TWT + \alpha(1-\lambda)EC$ where α is a scaling factor that relates the magnitude of the TWT and EC value to each other. We use $\alpha = TWT_{\max}/EC_{\max}$ where $TWT_{\max}$ is the TWT value obtained by list scheduling and the DTH scheme based on the earliest due date (EDD) rule and $EC_{\max}$ is the sum of $e(t)$ values over the entire scheduling horizon. On the one hand, the TWT value obtained by the EDD rule is large since this dispatching rule only relies on the due dates of the jobs and not on the weight. On the other hand, the $EC_{\max}$ value is also large enough since it is calculated on the assumption that no idle time is included in the schedule.

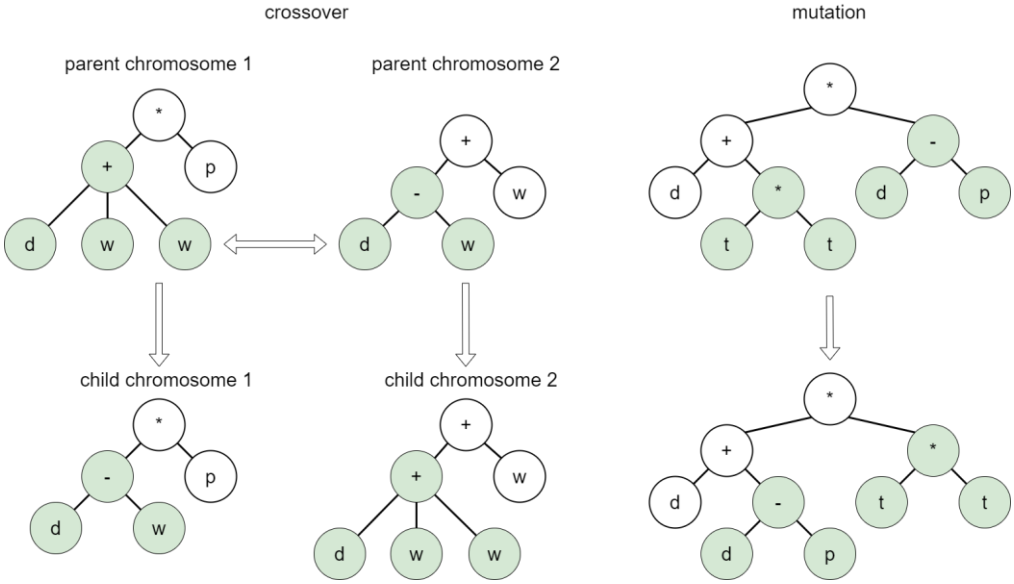

Figure 1: Crossover and mutation operators.

Table 1: Expression tree terminals.

Terminal	Description
d	due date of the current job
p	processing time of the current job
w	weight of the current job
t	current time
s	slack of the current job
ap	average processing time of all jobs
rp	average processing time of the remaining jobs
C	constant value from [0,9]
ec	$e(t)$ value of the current period
aec	average $e(t)$ value of all periods
rec	average $e(t)$ value of the remaining periods

The fitness value of the chromosome is obtained from the integrated objective function value. It is desirable to limit the depth of the expression trees and to allow only a maximum number of child nodes for each tree node. This avoids that only very specific dispatching rules are discovered.

An initial population is formed as follows. All terminals and functions from the Tables 1-2 have the same probability to be selected to form expression trees to be included into the initial population. A maximum initial depth of the trees is ensured during the generation process. Moreover, it is respected for

the initial population that only a maximum allowed number of child nodes is possible. Feasibility is ensured by repeating the random selection process until an appropriate number of child nodes is chosen and the depth of the trees is not too large. A dispatching rule is learned for a single problem instance or an entire set of problem instances for problem (1). This is called the training phase The overall GP approach, i.e. the training phase, is depicted in Figure 2.

Table 2: Set of functions.

Function	Description
$+$	$a + b$
$-$	$a - b$
$*$	$a * b$
$/$	a/b, if $b \neq 0$, 1 otherwise
H	$\max(a, b)$
L	$\min(a, b)$
$\wedge$	a^b
N	$-a$
EXP	$e^a = \exp(a)$

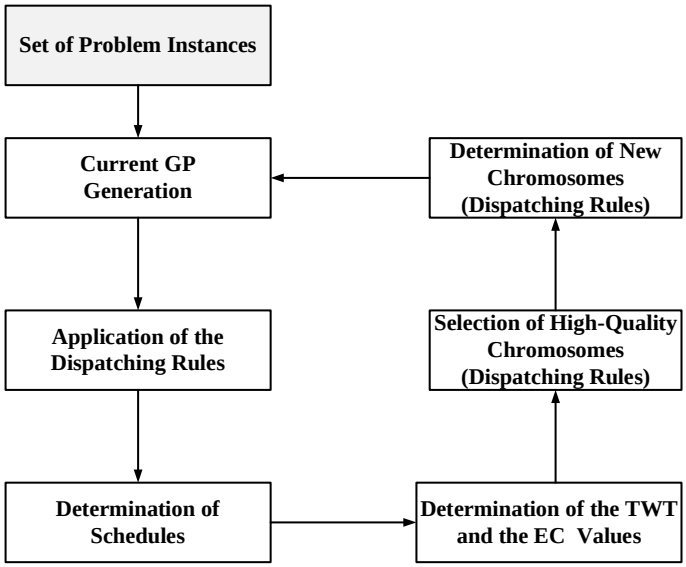

Figure 2: GP-based learning approach to discover new dispatching rules.

Afterwards, the specific dispatching rule is applied to problem instances with similar instance properties, for instance with respect to tightness of the due dates, than the instances used in the training phase. This application of the learned rule is called testing phase. This phase aims to assess the generalization capabilities of the learning scheme.

4 COMPUTATIONAL EXPERIMENTS

4.1 Design of Experiments

The performance of the discovered dispatching rules is assessed based on randomly generated problem instances. We expect that the performance of the learning scheme depends on the number of jobs n, the number of incompatible families F, the due date setting, namely the percent of tardy jobs T, the range of

the due dates R, and the maximum batch size B. The generation scheme for the instances is summarized in Table 3. Overall, we consider 2880 instances, 40 per factor combination. We use a TOU tariff appropriate for the winter time that consists of four time windows over the course of the scheduling horizon. Low, medium, and high EC values are assumed. We obtain:

$$e_W := \begin{cases} EC_h, & 1 \leq t < T/3 \\ EC_m, & T/3 \leq t < T/2 \\ EC_l, & T/2 \leq t < 5T/6 \\ EC_m, & \text{otherwise} \end{cases},$$

where EC_h, EC_m, and EC_l refer to a high, medium, and low EC value, respectively. Moreover, we use a summer tariff consisting of two time windows over the scheduling horizon. We have

$$e_s := \begin{cases} EC_h, & 1 \leq t < T/2 \\ EC_l, & \text{otherwise.} \end{cases}$$

We use 5, 10, and 20 instances from the same factor combination to learn a single rule in the training phase. The settings $\lambda \epsilon \{0.25, 0.75\}$ are used in the experiments.

A maximum computing time of 450, 900, and 1800s per GP run is used for 5, 10, and 20 instances applied in the training phase, respectively. The increasing amount of computing time covers the higher computational burden caused by a larger number of problem instances in the training phase. Here, the 5, 10, and 20 instances are always the first ones from the 40 independent instances for each factor combination. The last 20 instances per factor combination are used in the testing phase. Three independent GP runs are conducted per considered set of instances during the training phase. The resulting three dispatching rules are then applied in the testing phase.

Table 3: Instance generation scheme.

Factor	Level	Count
Number of jobs n	$n \in \{120,160,200\}$	3
Processing time of the jobs p_f	$p_j := \begin{cases} 2 \text{ with probability } p = 0.2 \\ 4 \text{ with probability } p = 0.2 \\ 10 \text{ with probability } p = 0.3 \\ 16 \text{ with probility } p = 0.2 \\ 20 \text{ with probability } p = 0.1 \end{cases}$	1
Weight of the jobs w_j	$w_j \sim U[0,1]$	1
Percentage of tardy jobs	$T \in \{0.3,0.6\}$	2
Range of the due dates	$R \in \{0.5,2.5\}$	2
Due date setting	$d_j \sim U[\mu(1-R/2), \mu(1+R/2)]$ $\mu := \hat{C}_{\max}(1-T)$ $\hat{C}_{\max} := n/(FB) \sum_{j=1}^{n_f} p_{f(j)}$	1
Number of job families F	$F \in \{2,4,6\}$	3
Maximum batch size B	$B \in \{4,8\}$	2
Factor combinations		72
Independent instances per factor combination		40
Total number of problem instances		2880

4.2 Parameterization and Implementation Issues

We use the following parameters shown in Table 4 in the GP scheme. They are determined following a trial and error strategy based on preliminary computational experiments with a limited number of instances from Table 3.

Table 4: Parameter settings.

Parameter	Value
Population size	500
Crossover probability pc	0.80
Mutation probability pm	0.05
Replacement rate rr	0.50
Maximum initial depth of a tree	8
Maximum depth of a tree	12
Maximum number of child nodes	4

The look-ahead parameter κ in the ATC rule is taken from the grid $\kappa = 0.1k$, where $k = 1, \ldots, 50$. The κ value that leads to smallest TWT value is used for comparison with the GP approach. We set the length of the scheduling horizon by estimating the makespan by

$$\hat{C} \coloneqq \left\lceil (1+\gamma) 1/B \sum_{j=1}^{n} \left(n_{f(j)} + 1\right) p_{f(j)} \right\rceil$$

where $\gamma = 0.80$ is a safety factor. All algorithms are coded in the C++ programming language. The GP is implemented using the GaLib framework (Wall 2022). The computational experiments are carried out on a PC with an Intel Core i5-6400 processor with 2.70 GHz and 16 GB of RAM.

4.3 Computational Results

In the experiments, the relative error of the GP approach with respect to the BATC-DTH in %

$$Imp \coloneqq \left((\lambda TWT + (1-\lambda)EC)(GP)/(\lambda TWT + (1-\lambda)EC)\,(BATC - DTH)\right)100\%$$

is reported. It is shown for all factor combination in Table 5 when 5 instances are simultaneously used in the training phase.

The abbreviation CT is used for the computing time of a single GP run. We present first the Imp values for the training phase, followed by the value for the testing phase after the slash. Improvements of the GP over the BATC-DTH are marked bold. Instead of comparing problem instances individually, the problem instances are grouped according to factor level values. We always report the average (Avg.), maximum (Max.), and minimum (Min.) values among comparable problem instances.

The corresponding results for 10 instances used in the training phase are reported in Table 6. We show again the results for the training and also the testing phase.

The corresponding results for 20 instances used in the training phase are reported in Table 7. The presentation of the results is the same as the one found in Tables 5 and 6. Due to space limitations, we show the results for the summer tariff only in an aggregated way in Figure 3. The results for 5, 10 and 20 training instances used in a single GP run are reported.

4.4 Analysis and Interpretation of the Results

We see from the Table 5 – 7 that the learned dispatching rules are on average slightly better than the BATC-DTH which can be considered as a rule that leads to high-quality schedules especially when the due dates

are tight (Mehta and Uzsoy 1998, Almeder and Mönch 2011). We observe from the tables that improvements up to 8.82% are possible for the training phase. Especially the factor combinations with $T = 0.3$ lead to the largest improvements. This is caused by the fact that the ATC rule does not work well in this situation when the number of tardy jobs is fairly small (Almeder and Mönch 2011). In the testing phase, improvements up to 10.29% are possible.

The setting $\lambda = 0.75$ usually leads to larger improvements of the GP approach over the BATC-DTH than $\lambda = 0.25$ for both the training and the testing phase. This observation can be explained by the fact that the room for improvement is smaller when the TWT measure is less important. We also see that $B = 8$ leads to larger improvements than $B = 4$. The same is true for a large number of incompatible families, i.e. $F = 6$.

Table 5: Computational results for the winter tariff for 5 instances and CT=450s per GP run.

Factor/Level	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**
	$\lambda = 0.75$			$\lambda = 0.25$		
$n = 120$	**1.51/1.12**	**6.71/9.17**	-3.88/-6.16	**0.50/0.61**	**3.19/3.17**	-5.26/-2.75
$n = 160$	**1.24/1.36**	**6.62/6.36**	-5.98/-7.31	**0.62/0.41**	**4.45/2.12**	-1.51/-2.56
$n = 200$	**0.47/0.63**	**7.70/7.34**	-6.06/-11.14	**0.34/0.33**	**2.90/2.06**	-2.02/-2.30
$T = 0.3, R = 0.5$	**0.72/1.09**	**4.75/6.36**	-2.80/-5.10	-0.28/**0.37**	**0.83/1.21**	-5.26/-0.61
$T = 0.3, R = 2.5$	**2.70/2.41**	**7.70/9.17**	-6.06/-5.24	**0.77/0.51**	**4.45/3.17**	-2.02/-1.70
$T = 0.6, R = 0.5$	-0.95/-1.21	**1.67/1.80**	-5.98/-7,31	**0.70/0.50**	**1.91/1.71**	-1.09/-0.54
$T = 0.6, R = 2.5$	**1.81/1.85**	**7.54/6.15**	-2.19/-11.14	**0.77/0.42**	**3.19/2.12**	-1.98/-2.75
$F = 2$	**0.95/0.21**	**5.39/5.87**	-3.00/-11.14	-0.02/**0.03**	**1.49/1.01**	-5.26/-2.56
$F = 4$	**0.99/1.18**	**7.70/9.17**	-6.06/-5.72	**0.77/0.65**	**4.45/3.17**	-1.81/-1.89
$F = 6$	**1.27/1.71**	**7.54/7,24**	-5.98/-7.31	**0.72/0.67**	**3.19/2.41**	-2.02/-2.75
$B = 4$	**0.51/0.32**	**5.39/6.08**	-6.06/-11.14	**0.32/0.30**	**2.33/1.87**	-2.02/-2.30
$B = 8$	**1.63/1.75**	**7.70/9.17**	-5.45/-5.72	**0.65/0.59**	**4.45/3.17**	-5.26/-2.75
Overall	**1.07/1.04**	**7.70/9.17**	-6.06/-11.14	**0.49/0.45**	**4.45/3.17**	-5.26/-2.75

Table 6: Computational results for the winter tariff for 10 instances and CT=900s per GP run.

Factor/Level	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**
	$\lambda = 0.75$			$\lambda = 0.25$		
$n = 120$	**0.90/1.15**	**7.23/10.29**	-9.15/-9.96	**0.51/0.70**	**3.29/2.61**	-4.90/-1.96
$n = 160$	**1.19/1.53**	**8.33/7.99**	-6.36/-4.28	**0.76/0.63**	**3.49/2.06**	-0.91/-0.85
$n = 200$	**0.38/1.03**	**6.90/6.01**	-7.11/-6.96	**0.49/0.42**	**4.10/2.46**	-1.35/-1.72
$T = 0.3, R = 0.5$	-0.17/**1.04**	**5.87/5.17**	-9.15/-9.96	-0.05/**0.29**	**0.72/1.01**	-1.35/-1.00
$T = 0.3, R = 2.5$	**3.10/2.63**	**8.33/10.29**	-4.67/-5.34	**1.06/0.82**	**4.10/2.61**	-4.90/-1.96
$T = 0.6, R = 0.5$	-0.87/-0.76	**1.81/1.48**	-4.97/-4.77	**0.60/0.52**	**1.99/1.63**	-0.88/-0.82
$T = 0.6, R = 2.5$	**1.24/2.03**	**4.75/6.30**	-8.41/-9.95	**0.74/0.71**	**3.01/2.06**	-0.76/-1.58
$F = 2$	**0.83/0.72**	**5.87/5.17**	-5.80/-9.96	**0.26/0.26**	**1.75/1.68**	-1.35/-1.72
$F = 4$	**0.33/1.33**	**5.46/10.29**	-8.41/-9.95	**0.45/0.67**	**3.49/2.61**	-4.90/-1.96
$F = 6$	**1.31/1.65**	**8.33/7.55**	-9.15/-5.77	**1.05/0.83**	**4.10/2.41**	-0.76/-1.19
$B = 4$	**0.15/0.28**	**7.23/6.10**	-9.15/-9.96	**0.31/0.37**	**3.29/1.79**	-4.90/-1.96
$B = 8$	**1.49/2.19**	**8.33/10.29**	-6.36/-3.74	**0.86/0.80**	**4.10/2.61**	-1.35/-1.19
Overall	**0.82/1.24**	**8.33/10.29**	-9.15/-9.96	**0.59/0.58**	**4.10/2.61**	-4.90/-1.96

Table 7: Computational results for the winter tariff for 20 instances and CT=1800s per GP run.

Factor/Level	Avg.	Max.	Min.	Avg.	Max.	Min.
	$\lambda = 0.75$			$\lambda = 0.25$		
$n = 120$	**1.29/1.64**	**8.82/10.08**	-6.30/-7.83	**0.60/0.83**	**3.86/3.03**	-2.97/-2.54
$n = 160$	**1.26/1.98**	**7.15/8.78**	-4.44/-4.45	**0.74/0.66**	**2.54/2.37**	-0.71/-1.86
$n = 200$	**0.19/0.89**	**5.67/7.36**	-6.79/-9.64	**0.55/0.63**	**2.62/2.91**	-1.58/-1.48
$T = 0.3, R = 0.5$	-0.11/**1.41**	**4.78/6.41**	-6.79/-7.83	**0.05/0.36**	**1.03/1.09**	-0.71/-1.86
$T = 0.3, R = 2.5$	**3.53/3.32**	**8.82/10.08**	-2.71/-1.92	**1.09/1.07**	**3.86/3.03**	-2.31/-1.48
$T = 0.6, R = 0.5$	-1.22/-1.02	**1.32/1.44**	-4.44/-4.45	**0.73/0.67**	**1.81/1.90**	-0.71/-0.65
$T = 0.6, R = 2.5$	**1.44/2.29**	**5.17/5.70**	-6.76/-9.64	**0.64/0.72**	**2.62/2.37**	-2.97/-2.54
$F = 2$	**0.68/0.96**	**5.43/4.86**	-6.79/-5.82	**0.31/0.31**	**1.84/1.93**	-0.81/-1.86
$F = 4$	**0.78/1.78**	**6.95/10.08**	-6.76/-9.64	**0.55/0.86**	**2.54/2.91**	-2.31/-2.02
$F = 6$	**1.27/1.76**	**8.82/7.30**	-6.30/-7.83	**1.03/0.95**	**3.86/3.03**	-2.97/-2.54
$B = 4$	**0.57/0.82**	**8.82/6.34**	-6.76/-9.64	**0.42/0.53**	**2.43/1.98**	-2.31/-2.02
$B = 8$	**1.25/2.18**	**8.11/10.08**	-6.79/-5.82	**0.84/0.87**	**3.86/3.03**	-2.97/-2.54
Overall	**0.91/1.50**	**8.82/10.08**	-6.79/-9.64	**0.63/0.70**	**3.86/3.03**	-2.97/-2.54

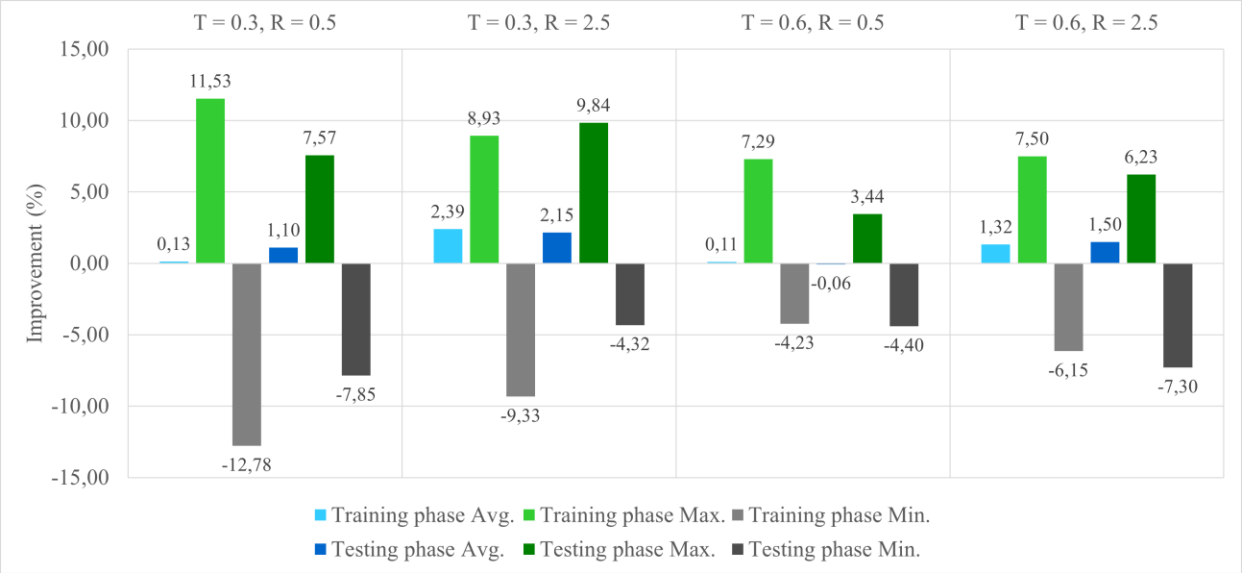

Figure 3: Aggregated computational results for the summer tariff.

Moreover, the discovered rules are able to generalize, i.e., they perform well beyond the limits of the training instance set. We also observe that more instances for training purposes increase the generalization ability of the proposed method. Regarding the difference of winter and summer tariff, the GP performs slightly better for the summer than for the winter tariff (see Figure 3 and the Tables 5-7).

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this paper, we studied an energy-aware scheduling problem for a single batch processing machine. A TOU tariff was assumed. The TWT and EC measures were considered within an integrated performance measure. A GP approach was designed for this problem. Computational experiments demonstrated that the

GP approach outperforms often a conventional list scheduling approach based on the BATC rule combined with the DTH scheme.

There are several directions for future research. First of all, it is interesting to include unequal release dates into the scheduling problem at hand. It is also interesting to extend the approach for more sophisticated machine environments such as parallel machines or flexible flow shops. Moreover, we are interested in using parallel programming techniques to speed up the population-based GP procedure.

ACKNOWLEDGMENTS

This research was partially supported by the MaxFab project that is funded by the University of Hagen. The authors gratefully acknowledge this financial support.

REFERENCES

Almeder, C., and L. Mönch. 2011. "Metaheuristics for Scheduling Jobs with Incompatible Families on Parallel Batch Machines." *Journal of the Operational Research Society* 62:2083–2096.

Fowler, J. W., and L. Mönch. 2022. "A Survey of Scheduling with Parallel Batch (p-batch) Processing." *European Journal of Operational Research* 298(1):1–24.

Geiger, C., R. Uzsoy, and H. Aytuk. 2006. "Rapid Modeling and Discovery of Priority Dispatching Rules: An Autonomous Learning Approach." *Journal of Scheduling* 9:7–34.

Geiger, C., and R. Uzsoy. 2008. "Learning Effective Dispatching Rules for Batch Processor Scheduling." *International Journal of Production Research* 46(6):1431–1454.

Graham R. L., E.L Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. 1979. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey." *Annals of Discrete Mathematics* 5:287–326.

Hildebrandt, T., D. Goswami, and M. Freitag. 2014. "Large-scale Simulation-based Optimization of Semiconductor Dispatching Rules." *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, pp. 2580-2590. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kuck, M., E. Broda, M. Freitag, T. Hildebrandt, and E. M. Frazzon. 2017. "Towards Adaptive Simulation-based Optimization to Select Individual Dispatching Rules for Production Control." *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, pp. 3852-3863. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lawler, E. L. 1977. A "Pseudopolynomial" Time Algorithm to for Sequencing Jobs on Minimize Total Weighted Tardiness." *Annals of Discrete Mathematics* 1:331–342.

Liu, C.-H. 2014. "Approximate Trade-off Between Minimisation of Total Weighted Tardiness and Minimisation of Carbon Dioxide (CO2) Emissions in Bi-criteria Batch Scheduling Problem." *International Journal of Computer Integrated Manufacturing* 27(8):579–771.

Mehta, S. V., and R. Uzsoy. 1998. "Minimizing Total Tardiness on a Batch Processing Machine with Incompatible Job Families." *IIE Transactions* 30 (2): 165–178 .

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd ed., Berlin: Springer.

Mönch, L., H. Balasubramanian, J. W. Fowler, and M. E. Pfund. 2007. "Heuristic Scheduling of Jobs on Parallel Batch Machines with Incompatible Job Families and Unequal Ready Times." *Computers & Operations Research* 32:2731–2750.

Mönch, L., J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. 2011. "A Survey of Problems, Solution Techniques, and Future Challenges in Scheduling Semiconductor Manufacturing Operations." *Journal of Scheduling* 14(6):583-599.

Mönch, L., J. W. Fowler, and S. J. Mason. 2013. *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems*. New York: Springer.

Mönch, L., R. Uzsoy, and J. W. Fowler. 2018. "A Survey of Semiconductor Supply Chain Models Part I: Semiconductor Supply Chains and Strategic Network Design". *International Journal of Production Research* 56(13):4524-4545.

Nguyen, S., Y. Mei, and M. Zhang. 2017. "Genetic Programming for Production Scheduling: A Survey with a Unified Framework". *Complex & Intelligent Systems* 3(1):41–66.

Rocholl, J., L. Mönch, and J. W. Fowler. 2020. "Bi-criteria Parallel Batch Machine Scheduling to Minimize Total Weighted Tardiness and Electricity Cost." *Journal of Business Economics* 90:1345–1381.

Singapore Government. 2022. "Energy Efficiency in the Microelectronics Industry." https://www.e2singapore.gov.sg/DATA/0/docs/NewsFiles/Energy%20efficiency%20in%20the%20microelectronics%20industry%20v2.pdf. Last accessed 23 April 2022.

T'kindt, V., and J.-C. Billaut. 2002. *Multicriteria Scheduling*. Berlin: Springer.

Uzsoy, R. 1995. "Scheduling Batch Processing Machines with Incompatible Job Families." *International Journal of Production Research* 33(10):2685–2708 .

Wall, M. 2022. "GaLib: A C++ Library for Genetic Algorithm Components." Documentation. https://lancet.mit.edu. Last accessed 23 April 2022.

Yu, C.-M., C.-F. Chien, and C.-J. Kuo. 2017. "Exploit the Value of Production Data to Discover Opportunities for Saving Power Consumption of Production Tools". *IEEE Transactions on Semiconductor Manufacturing* 30(4):345–350.

AUTHOR BIOGRAPHIES

DANIEL SASCHA SCHORN is a master student in information systems at the University of Hagen. He received a bachelor degree in Information Systems from the University of Hagen. His research interests are scheduling for semiconductor manufacturing. His e-mail address is dan.schorn@gmx.de.

LARS MÖNCH is full professor of Computer Science at the Department of Mathematics and Computer Science, University of Hagen where he heads the Chair of Enterprise-wide Software Systems. He holds M.S. and Ph.D. degrees in Mathematics from the University of Göttingen, Germany. After his Ph.D., he obtained a habilitation degree in Information Systems from Technical University of Ilmenau, Germany. His research and teaching interests are in information systems for production and logistics, simulation, scheduling, and production planning. His email address is Lars.Moench@fernuni-hagen.de. His website is https://www.fernuni-hagen.de/ess/team/lars.moench.shtml.