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.

SCHEDULING JOBS WITH UNCERTAIN READY TIMES ON A SINGLE BATCH PROCESSING MACHINE

Jens Rocholl
Fajun Yang
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). A batch is a group of jobs that are processed at the same time on a single machine. The jobs belong to incompatible families. Only jobs of the same family can be batched together. Each job has a weight, a due date, and a ready time. The performance measure of interest is the total weighted tardiness (TWT). The ready times are calculated based on information related to upstream machines that is stored in a Manufacturing Execution System (MES). Therefore, they can be uncertain. We propose a genetic algorithm (GA)-based approach. Sampling is used to take into account the uncertainty of the ready times. Results of computational experiments are reported that demonstrate that this approach performs well with respect to computing time and solution quality.

1 INTRODUCTION

Dispatching is an important production control function in wafer fabs (Mönch et al. 2013). However, with the recent dramatic increase in computer efficiency, deterministic scheduling approaches have become more competitive (Mönch et al. 2011). Scheduling in wafer fabs is challenging due to the complicated process conditions such as unrelated parallel machines with dedications, sequence-dependent setup times, and batch processing machines. A batch is a group of jobs that are processed at the same time on the same machine. We refer to this setting as parallel batching, for short p-batching. Up to one third of all operations in wafer fabs are carried out on batch processing machines (Fowler and Mönch 2022). The processing time of operations on batch processing machines are often long, up to one day, compared to the processing time of operations on regular machines. Hence, careful dispatching and scheduling decisions are crucial for wafer fab performance. Scheduling decisions for batch processing machines require in addition to conventional assignment and sequencing decisions grouping decisions which determine which job should belong to a batch. However, a batch can only start with processing if all jobs that belong to it are ready for processing. Therefore, on the one hand, precise information about the ready times is crucial for batching decisions. On the other hand, the ready time of a job can be only determined based on information for operations of the job on upstream machines found in the MES. Hence, this information is often uncertain.

In the present paper, we make a first attempt to consider uncertain ready times in a scheduling problem for a single batch processing machine. This problem is clearly a simplified model problem that does not cover the full set of process restrictions that are common for scheduling problems for work centers with batch processing machines, such as the diffusion work center in wafer fabs (Yugma et al. 2012). However,

it allows us to understand main principles, mainly sampling techniques, for dealing with the uncertainty of ready times in batch scheduling problems.

The paper is organized as follows. In the next section, we describe the problem and discuss previous work. We then present heuristic scheduling approaches in Section 3. The results of computational experiments are discussed in Section 4. Conclusions and future research directions are summarized in Section 5.

2 SINGLE-MACHINE BATCH SCHEDULING PROBLEM

2.1 Problem Statement

We consider a single batch processing machine with maximum batch size B measured in number of jobs. The n jobs to be scheduled belong to incompatible families $f = 1, \ldots, F$. Only jobs of the same incompatible family can be batched together. There are n_f jobs in family f. Each job has a due date d_j, a ready time r_j, and a weight w_j indicating the importance of the job. The processing time of a batch belonging to family f is p_f whereas the processing time of job j is $p_{f(j)}$ where $f(j)$ indicates the family of the job. This means that all jobs of the same family have an identical processing time. The performance measure of interest is the TWT defined 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 deterministic version of the problem at hand can be stated as follows:

$$1|p - batch, incompatible, r_j|TWT, \tag{1}$$

where $p - batch$ indicates parallel batching and $incompatible$ refers to the incompatible job families.

We assume that all data of problem (1) except the ready times r_j are known in advance and deterministic. For the ready times, however, we assume as known the probability distribution of the delays relative to the ready time values found in the MES. We refer to this problem as the stochastic counterpart (SCP) of problem (1). We are interested in determining a schedule for the SCP that leads to a small expected TWT value when the schedule is executed under uncertainty. Following Sevaux and Sörensen (2004), we call the resulting schedules robust with respect to quality. For a given problem instance of the SCP of problem (1), we solve problem (1) using the ready times found in the MES. Moreover, we also consider ready times that incorporate the expected value of the uncertain delays.

2.2 Related Work

There are only a very few papers that deal with p-batch scheduling problems involving uncertain data (Fowler and Mönch 2022). The two rare exceptions are Shahnaghi et al. (2016a), (2016b) where a real-world flow-shop scheduling problem with batch processing machines is studied. The processing times and the unequal job sizes are uncertain. First, a mixed integer linear programming approach for the deterministic counterpart is established. Moreover, two robust formulations are provided along with a particle swarm optimization approach.

Next, we discuss Monte Carlo sampling approaches that are able to deal with uncertain problem data in scheduling problems. A real-world flow shop scheduling problem with a finite buffer between stages is solved by Almeder and Hartl (2014). They use variable neighborhood search (VNS). Discrete-event simulation is applied to assess the quality of the applied moves. Throughput and buffer utilization are used as performance measures. A flexible job-shop scheduling problem with stochastic processing times is studied by Gomez et al. (2021). They use a tabu search approach to maximize the probability of the makespan to be smaller than a prescribed value. Sampling, i.e. Monte Carlo simulation, is used to evaluate the quality of the moves.

The most pertinent paper is Sevaux and Sörensen (2004). A single-machine scheduling problem is considered. The performance measure is the weighted number of tardy jobs. The ready times are uncertain. A GA is proposed to solve this problem. Sampling is used to assess the chromosomes under uncertainty. A similar sampling approach is also applied by Sevaux and Sörensen (2009) for vehicle routing problems with

uncertain travel times. In the present paper, we apply the sampling technique of these two papers. However, the scheduling problem at hand is different from the scheduling problem considered by Sevaux and Sörensen (2004) since we have to make batching decisions which are grouping decisions by nature. We will see that batching decisions require repair strategies during execution under uncertainty to obtain high-quality robust schedules.

3 HEURISTIC APPROACHES FOR DETERMINISTIC AND UNCERTAIN READY TIMES

3.1 Heuristic for Deterministic Ready Times

A grouping GA (GGA) is designed to solve problem (1) and its SCP. This population-based heuristic iteratively produces generations of solutions until a predefined stopping criterion is met. GGAs in particular are proposed to cope with problems with grouping characteristic, motivated by the observation that conventional encoding schemes are not well-suited for this type of problems (Falkenauer 1996). With a GGA the genes of a chromosome represent the grouping entities, i.e. batches. Genetic operators of the GGA are specifically designed to preserve favorable grouping decisions.

The crossover operator is adapted to take into account incompatible families as proposed by Sobeyko and Mönch (2011). The operator works as follows. First, two crossing sections are randomly chosen for each of the two parent chromosomes. Then, the demarcated part of the chromosome of the first parent is deleted and replaced by the selected part of the second one. The resulting interim child chromosome may contain certain jobs twice while other ones are missing. Duplicate jobs are removed from those batches inherited from the first parent. Eventually, missing jobs need to be inserted. This reinsertion can significantly impact the performance of a GGA (cf. Brown and Sumichrast 2003). In the present paper, it is implemented in such a way that jobs are inserted into the batches that are processed as earliest, with start times equal to or larger than the jobs´ ready times. The crossover operation is illustrated by means of an example in Figure 1.

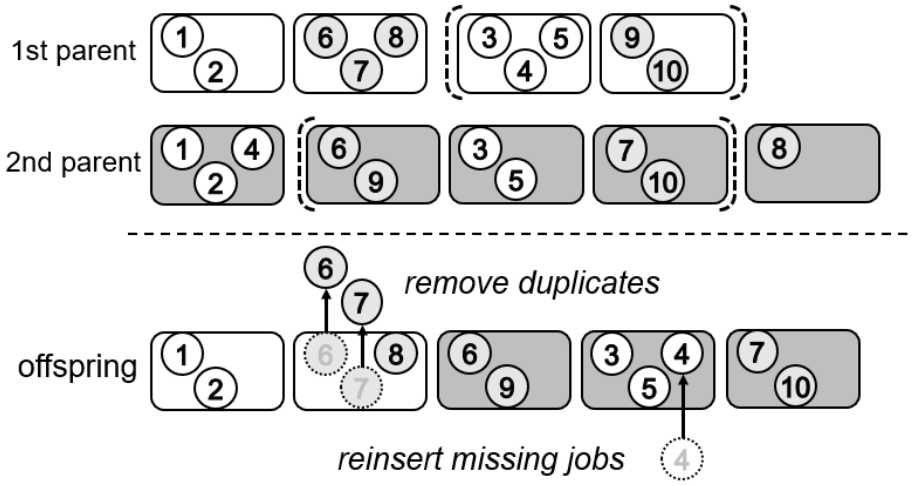

Figure 1: GGA crossover operation.

Jobs are represented by circles and labeled by their indices. The jobs with indices 1-5 belong to family 1. Jobs with indices 6-10 belong to family 2. Batches are represented by rectangles. Different shades indicate from which parent a batch is inherited. Crossing sections are marked by dashed brackets. The interim offspring inherits the jobs 6 and 7 from both parents. Consequently, duplicates are removed from the second batch (inherited from the first parent). Job 4 is passed on by neither parent. It must therefore be reinserted into the offspring. It could be assigned to either one of the two batches with jobs belonging to

the first family. In this example we assume the ready time of job 4 is larger than the start time of the batch containing the jobs 1 and 2. According to the chosen reinsertion strategy it is therefore inserted into the batch containing the jobs 3 and 5.

For scheduling problem (1), we add a random key to each gene to encode the batch sequence by sorting the random keys. A chromosome can be mutated with a given mutation probability. In this case, one of the chromosome´s batches is randomly chosen, and the mutation operator randomly resets its random-key value and thus possibly changes its position within the sequence. Roulette wheel selection is used to choose parents for recombination.

Previous applications of GGAs (Sobeyko and Mönch 2011, Rocholl et al. 2020) show that their performance greatly profits from local search. For each chromosome obtained by initialization or crossover a set of local search operations is executed. First, a single pass through the set of batches ordered by their assigned position is carried out to avoid unnecessary idle times. Each batch is rescheduled to be executed as soon as possible as long as the already rescheduled batches are not affected. Thus, gaps in the original schedule can be filled with batches available earlier. With the second pass through the batches from the back to the front of the schedule, each job is inserted into the first batch with a start time equal or larger than the job´s ready time, sufficient available capacity, and a matching family. The jobs are considered in the order of non-increasing w_j/d_j values for each batch. After performing theses deterministic operations, the following five types of randomized moves can be carried out if they lead to an improvement of the TWT value:

1. Randomly chosen single jobs can be inserted to randomly chosen batches of the same family.
2. Two randomly chosen jobs from different batches of the same family can change positions.
3. A randomly chosen batch can be inserted at a randomly chosen position.
4. Two randomly chosen batches can exchange positions.
5. Finally, a randomly chosen batch can be split into two batches.

The number of attempted moves is limited to ten for each neighborhood to reduce the computational burden. Chromosomes of the initial population are generated randomly. Therefore a single pass through the set of jobs assigns the jobs to batches. For each job, one of the batches is randomly chosen with equal probability. If the job cannot be assigned to the chosen batch because there are already jobs belonging to a different family assigned or there is not enough available capacity, another batch is randomly chosen until the job is eventually assigned to some batch. Random keys are set to define the batch sequence. Finally local search is performed on each chromosome.

3.2 Sampling-based Heuristic for Uncertain Ready Times

For the deterministic configuration of the proposed algorithm, whenever the TWT value of a schedule is computed or whenever the effect of a local search move is assessed, ready times from the MES, eventually modified by the expected value of the delay, are considered. It is thus implicitly assumed that a schedule can be executed as intended.

However, this is not the case for the SCP of problem (1) as delayed job releases will cause delayed processing of batches. For computing robust schedules, the conventional fitness function is therefore replaced by a robust evaluation function to proactively incorporate uncertain problem data (cf. Sörensen 2001). The distribution of the delays of the ready times is assumed to be known. A set of sample values is created with each sample containing realizations of the corresponding random variables. Prior to the evaluation of a schedule, it is adapted to each sample, possibly yielding a different adapted schedule for each sample. The robust evaluation function then returns an estimate for the TWT value of the adapted schedules by computing the average of the TWT values of the samples. The number of samples can be expected to have a contrary influence on the capability to suitably represent the actual manifestation of

parameters and the required computing time. Preliminary computational experiments indicate that 20 samples are sufficient to evaluate a schedule under uncertainty.

3.3 Repair Strategies for Executing the Schedule Under Uncertainty

Schedules computed based on uncertain ready times often cannot be executed as originally intended. In the case of uncertain ready times, some jobs will become available later than expected and thus delay the processing of related batches, eventually resulting in delays of subsequent batches. In practice, one would react to such disturbances by adapting the schedule to the current situation. Different rescheduling policies can be employed to mitigate the negative effects of delayed job releases.

Different rescheduling policies (RSP) to alter the batch formation are proposed. If an actual release of a job would lead to a delay of the start of the associated batch then the job is removed from that batch. The policies are as follows:

1. The simplest approach assigns the removed job to the first succeeding batch with jobs of the same family, with a start time equal to or larger than the actual ready time of the job and sufficient available capacity. An advantage of this policy is that only jobs with ready times later than expected need to be considered whereas all other jobs can remain in place. Therefore, on the one hand, only a few batches are affected. On the other hand, especially for schedules with tightly packed batches, important jobs could be moved from early positions to the end of the schedule and thus have a high impact on the TWT value. We refer to this strategy as RSP1.
2. With the second policy, again a delayed job is moved to the earliest succeeding batch of the same family and with a start time not smaller than the actual ready time of the job. If that batch is full, room is made available by removing the job with the smallest w_j/d_j value. The removed job is in turn assigned to the next succeeding batch. This step can be repeated. In tightly packed schedules this can imply a cascade of reassigning jobs to subsequent batches while each job´s individual delay rests rather small. This strategy is called RSP2.
3. The third approach works like the RSP2, but each time a job is removed from a batch the succeeding batches are searched for an available job to fill in the gap. The first job to be found possible for being pulled forward is inserted, i.e., the job is left-shifted and its contribution to the TWT value of the schedule is possibly reduced. If more than one job from that batch could be pulled forward, ties are broken by preferring the job with the largest w_j/d_j value. We abbreviate this strategy by RSP3.

Note that both the RSP1 and RSP2 only consider right shifting jobs, whereas RSP3 also allows a left-shift of jobs.

4 COMPUTATIONAL EXPERIMENTS

4.1 Design of Experiments

The GGA and the repair strategies are assessed based on randomly generated problem instances. The design of experiments is inspired by similar research with uncertain ready times by Sevaux and Sörensen (2004) which is adapted to take into account the batch scheduling problem. We expect that the performance depends mainly on the maximum batch size B. Therefore, the design of experiments summarized in Table 1 is used. Here, $gamma(\alpha, \beta)$ refers to a gamma distribution with shape parameter $\alpha > 0$ and scale parameter $\beta > 0$. Moreover, $DU[a, b]$ refers to a discrete uniform distribution over the set of integers $\{a, \ldots, b\}$. In a first set of experiments, we are interested in testing if the advantage of a sampling approach observed by Sevaux and Sörensen (2004) for similar problems carries over to scheduling problems with the TWT measure. Therefore, a set of instances of the special case of problem (1) and the SCP with $B = 1$ from Table 1 is considered. A scheduling horizon with length $T := 10n$ is introduced with the purpose of deriving ready times and due dates. The ready times are distributed according to the distribution $gamma(4, T/16)$ with mean $0.25T$ and a standard deviation of $0.125T$.

As already stated in Subsection 3.3, it is likely from a practical point of view that batch formation decisions will be revised in the face of delayed jobs during execution. Instead schedules are adapted, i.e., delayed jobs are reassigned to not hinder a batch from being started on time. Therefore, we perform a second set of experiments where we investigate the impact of the rescheduling strategies for schedules that are computed based on deterministic ready times and executed under uncertainty.

Table 1: Design of experiments.

Factor	Level	Count
Number of families F	4	1
Number of jobs per family n_f	15	1
Maximum batch size B	1, 2, 4, 8	4
Processing time p_f	$p_f \sim DU[5, 20]$	1
Scheduling horizon T	$T := 10n$ for $B = 1$ $T := 5n$ for $B = 2$ $T := 3n$ for $B = 4$ $T := 2n$ for $B = 8$	1
Job ready time r_j	$r_j \sim gamma(4, T/16)$	1
Job due date d_j	$T - d_j \sim gamma(4, T/16)$	1
Job weight w_j	$w_j \sim DU[1, 10]$	1
Number of independent replications	10 per factor combination	10
Total number of problem instances		40

In a last series of experiments, one of the proposed rescheduling policies RPS1-RPS3 is incorporated into the GGA to enable it to find adaptable schedules. Whenever the robust evaluation function requires a schedule to be adapted to a sample, the chosen policy is executed before. It is of course mandatory to also apply the rescheduling policy when schedules obtained by both the deterministic and robust GGA configuration are evaluated and compared under uncertainty. To assess the influence of different repair strategies in the case of batching, i.e. for $B > 1$, the instances of Table 1 are solved to assess the performance of the proposed robust heuristic with the sampling method incorporated against the conventional GGA based on deterministic data. An actual realization of such an instance is obtained by first randomly choosing 20% of the jobs which can be delayed. For each of these jobs a delay taken from $DU[0, \lceil l\bar{p} \rceil]$ is then added to the ready time from the MES where $\bar{p}$ is the average processing time of all jobs and l is a parameter to represent the magnitude of possible delays. Three levels of uncertainty represented by $l \in \{0.5, 1, 2\}$ are used in the computational experiments.

As mentioned above, the probability distribution of delays is assumed to be known in advance. It therefore seems reasonable to consider ready times with the added expected value of the delay when the conventional GGA with deterministic data is used. Consequently, expected ready times are computed by $r_j^{exp} := r_j + 0.1\lceil l\bar{p} \rceil$. However, it is impossible to predict which 20% of the jobs will be delayed. Thus, the expected ready times overestimate the delay for 80% of the jobs, which may affect the algorithm´s ability to find near-optimal schedules. For this reason, the deterministic GGA is performed with both expected and optimistic ready times $r_j^{opt} := r_j$, where the delay is zero for all jobs.

4.2 Parameterization and Implementation Issues

The population size of the GA is set to 50. The percentage of the population to be replaced with each new generation is 80%. A crossover probability of 0.9 and a mutation probability of 0.05 are set. These parameter values are determined by preliminary computational experiments using a trial and error strategy based on a small subset of instances from Table 1. The algorithm terminates when 100 consecutive

generations with no improvement of the (expected) TWT value are performed. Because of the stochastic nature of the GGA, each problem instance is solved five times using the conventional and the robust GGA with a different seed to obtain statistically significant results. Each pair of schedules is then executed under uncertainty with 20 samples of the ready times, and the TWT values are computed using the robust evaluation function. Of course, the set of samples used during the solution phase of the robust GGA is different from the sample set used during the evaluation phase, i.e., when the schedule is executed under uncertainty. This is important to allow for a fair comparison with actual uncertain ready times. The GGA is coded in the C++ programming language using the GaLib framework (Wall 2022). All experiments are conducted on a computer with Intel Core i7-2600 CPU with 3.40 GHz and 16 GB RAM.

4.3 Computational Results

Instead of reporting the average TWT values per instance, Table 2 reports the average over the ten instances of each factor combination in Table 1. The average TWT values of a baseline heuristic, the reference R, are considered. The heuristic ratio $HR_H := TWT_H/TWT_R$ for a heuristic H is computed. Here, we use the deterministic and robust configuration of the GGA. Results of the first experiment instances for $B = 1$ are shown in Table 2. The columns named "HR_{det}" and "HR_{rob}" show the results of the deterministic variant of the GGA and the robust one, respectively, while the column "Diff." show the relative difference. Negative values indicate advantages of the robust approach.

Table 2: Results for instances of the special case $B = 1$.

Delay	Ready times	HR_{det}	HR_{rob}	Diff.
[0, 0.5$\bar{p}$]	optimistic	1.000	0.997	-0.3%
	expected	1.000	0.991	-0.9%
[0, $\bar{p}$]	optimistic	1.000	0.986	-1.4%
	expected	1.000	0.979	-2.1%
[0, 2$\bar{p}$]	optimistic	1.000	0.958	-4.2%
	expected	1.000	0.963	-3.7%

Results from experiments to compare different rescheduling policies for instances with $B > 1$ are visualized with a bar chart in Figure 2. Each triplet of bars represents the average TWT value of schedules executed under uncertainty with no repair, RSP1, and RSP3 of ten instances with a given maximum batch size. Three different levels of uncertainty are considered.

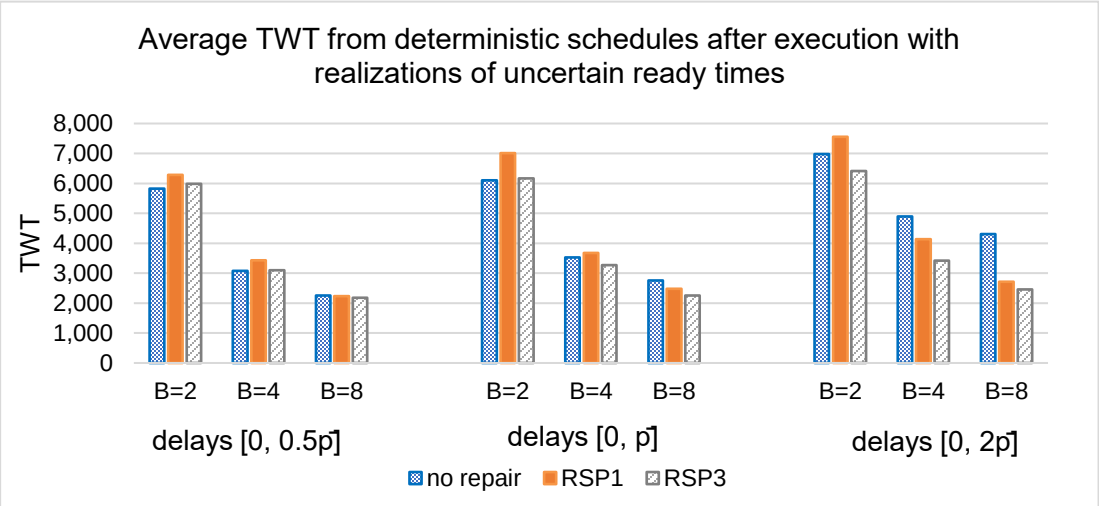

Figure 3: TWT values obtained by different rescheduling policies.

Results of the final experiment to compare the deterministic and robust variants of the GGA with batching are shown in Table 3.

We present results from the situation where no rescheduling is applied, i.e., all batches remain formed as intended and are started as soon as all assigned jobs are available, along with the policies RSP1 and RSP3. As the RSP3 is basically just an improved variant of the RSP2, results of experiments with the RSP2 are not reported for the sake of brevity. Note that the column entitled "Diff." indicates the relative difference between the results obtained by the deterministic and the robust GGA when the same rescheduling policy is employed.

Table 3: Results from experiments with batching and optimistic ready times.

		w/o rescheduling			**RSP1**			**RSP3**		
Delay	**B**	**HR_{det}**	**HR_{rob}**	**Diff.**	**HR_{det}**	**HR_{rob}**	**Diff.**	**HR_{det}**	**HR_{rob}**	**Diff.**
[0, 0.5$\bar{p}$]	2	1.000	1.019	1.9%	1.078	1.053	-2.4%	1.028	1.037	0.9%
	4	1.000	1.040	4.0%	1.113	1.072	-3.7%	1.004	1.022	1.8%
	8	1.000	0.988	-1.2%	0.992	0.995	0.3%	0.964	0.959	-0.5%
[0, $\bar{p}$]	2	1.000	1.001	0.1%	1.149	1.048	-8.9%	1.010	0.996	-1.4%
	4	1.000	0.970	-3.0%	1.045	0.991	-5.2%	0.930	0.905	-2.7%
	8	1.000	0.984	-1.6%	0.900	0.866	-3.7%	0.820	0.791	-3.6%
[0, 2$\bar{p}$]	2	1.000	0.978	-2.2%	1.083	0.986	-8.9%	0.919	0.904	-1.6%
	4	1.000	0.935	-6.5%	0.845	0.819	-3.1%	0.698	0.684	-2.0%
	8	1.000	1.049	4.9%	0.629	0.598	-4.8%	0.571	0.546	-4.4%
Average		1.000	0.995	-0.5%	0.995	0.942	-5.3%	0.887	0.876	-1.2%

Table 4: Results from experiments with batching and expected ready times.

		w/o rescheduling			**RSP1**			**RSP3**		
Delay	**B**	**HR_{det}**	**HR_{rob}**	**Diff.**	**HR_{det}**	**HR_{rob}**	**Diff.**	**HR_{det}**	**HR_{rob}**	**Diff.**
[0, 0.5$\bar{p}$]	2	1.000	1.008	0.8%	1.073	1.043	-2.9%	1.014	1.029	1.4%
	4	1.000	1.032	3.1%	1.082	1.067	-1.4%	1.037	1.018	-1.9%
	8	1.000	0.971	-2.9%	0.997	0.975	-2.3%	0.995	0.937	-6.1%
[0, $\bar{p}$]	2	1.000	0.983	-1.7%	1.093	1.033	-5.8%	1.007	0.982	-2.6%
	4	1.000	0.950	-5.3%	1.037	0.967	-7.3%	0.944	0.878	-7.4%
	8	1.000	0.946	-5.7%	0.899	0.833	-7.9%	0.859	0.766	-12.1%
[0, 2$\bar{p}$]	2	1.000	0.979	-2.1%	1.107	0.986	-12.3%	0.944	0.901	-4.8%
	4	1.000	0.960	-4.2%	0.935	0.852	-9.7%	0.830	0.716	-15.9%
	8	1.000	1.050	4.8%	0.725	0.599	-20.9%	0.729	0.545	-33.8%
Average		1.000	0.987	-1.5%	0.994	0.928	-7.8%	0.929	0.863	-9.2%

4.4 Analysis and Interpretation of the Results

Results from the first set of experiments for the special case of $B = 1$ are shown in Table 2. Obviously, for this special case we can indeed observe a consistent benefit for robust schedules, although admittedly this

is fairly low for small delays. Even more important, the advantages of robust schedules are consistently demonstrated for each single problem instance (not shown in detail). No significant differences between assuming optimistic ready times or ready times with added expected delay are observed.

A comparison of the results for schedules obtained by the deterministic GGA without repairing versus the ones with RSP1 and RSP3 is reported in Figure 2. This experiment is based on optimistic ready times. We are interested in assessing the benefit resulting from the application of each policy. Obviously higher advantages of rescheduling can be observed in face of higher delays and a larger maximum batch size. One can also state the RSP3 always outperforms RSP1, which is not surprising as this policy is the more elaborate one. On average, the application of RSP3 yields savings of around 11% compared to not using rescheduling at all. A rather counterintuitive observation is made for the instances with B=2 and also B=4 with RSP1. Apparently, in those cases rescheduling does in fact not affect the results. A likely explanation is that in the case of small maximum batch sizes, it may often be less disruptive to postpone the few jobs of the entire batch rather than splitting batches. It is also less likely to find available capacity in batches to fill in relocated jobs. An appropriate way to mitigate this disadvantage could be setting a threshold value for the delay below which the job would remain in place. More sophisticated reactive procedures could be designed but this is beyond the scope of the present paper.

Next, we discuss the results of the third set of experiment shown in Tables 3 and 4. In the beginning of this research, the focus was clearly on the application of sampling as a proactive technique to find robust schedules. Sampling techniques had already been successfully applied for scheduling problems (cf. Sevaux and Sörensen 2004). We were therefore expecting the robust configuration of the GGA to be capable to find superior solutions compared to the deterministic one in an uncertain setting. However, as can be seen in the first columns of Table 3, if rescheduling is omitted there is no consistent pattern on what configuration yields better results. At first sight, it seems the conventional approach outperforms the robust one and vice versa. But in the case of large delays from the interval $[0, 2\bar{p}]$ and B=8 the outcome does not match this pattern, as schedules from the robust GGA even turn out to be less robust on average. The inconsistency can also be observed on the level of individual instances within a single factor combination (not displayed). Considering the fact that sampling generally comes along with larger computing times, we can state that the robust GGA is apparently not well-suited to handle uncertain ready times in the given scenario. As these observations are rather disappointing and quite in discrepancy with the results found for the special case of $B = 1$, it turns out that sampling without repair does not work well.

Apparently, batching is an additional challenge for the application of proactive robust techniques. This can be explained by the fact that delays of single jobs can directly affect a larger number of jobs assigned to the same batch. This directly leads to the question, if and how the heuristic has to be changed to provide more robust schedules for batching problems. The advantage of rescheduling policies shown before suggests that by incorporating these into the sampling method the heuristic may be able to find schedules which are, even though not immune to changes of ready times, at least more easily adaptable. To verify this assumption, we turn towards the results obtained by the GGA with the incorporated RSP1 and RSP3. We first analyze the results shown in Table 3. Contrary to the GGA without rescheduling, schedules obtained with including the RSP1 and the RSP3 can indeed be found more robust. Moreover, the results are more consistent. With RSP1, the robust schedules yield TWT values overall around 5.3% below the ones obtained without rescheduling. However, one has to admit that the RSP1 is less effective than the RSP3 and even sometimes affects the results compared to GGA without rescheduling. Therefore, the schedules found by the GGA with the RSP3 seem to be more interesting from a practical point of view. The overall savings of around 1.2% at first does not appear encouraging. Yet, disappointing results are mainly observed for small delays and small maximum batch sizes. As mentioned above, we are confident that the robustness can be improved by incorporating a threshold value into the rescheduling policy. The benefit of robust schedules is more significant for larger maximum batch sizes, i.e. between 2.7% and 4.4% for $B = 8$.

It should be pointed out that robustness in this context actually can be interpreted as adaptable with less deterioration. The major insight gained during the research is that for this type of p-batching problems proactive sampling methods should be combined with a suitable reactive rescheduling policy.

The results of the deterministic GGA shown in Table 4 are obtained assuming ready times with an expected value of the delay added. One can see that with no repairing strategy used, the objective values are up to 4% worse compared to when optimistic ready times are assumed. This disadvantage even dramatically increases for the cases of RSP1 and RSP3 being applied. The TWT values are impaired by up to 27%, leading to an advantage of almost 34% for the robust approach in one case. One must conclude, that in the given scenario with only 20% of the jobs´ ready times being actually delayed, the deterministic GGA is better used with optimistic ready times. Apparently, especially in the context of batching it is disadvantageous to overestimate ready times, and it also seems to hinder the repair strategies to work properly.

The benefits of employing sampling generally come at the cost of additional required computing time. With the given hardware and parameter settings, the GGA with the deterministic configuration requires an average of 20 seconds per instance to terminate, while the robust variant requires around 680 seconds.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In the present paper, we discussed a scheduling problem for a single batch processing machine with uncertain ready times of the jobs. The TWT performance measure was considered. We applied a GGA. A sampling technique was proposed to assess the fitness of the chromosomes by executing the schedule under uncertainty. We were able to compute high-quality robust schedules when appropriate repair strategies were incorporated during schedule execution under uncertainty.

There are several directions for future research. First of all, we believe that the proposed method can be extended towards p-batch scheduling problems for more complicated machine environments such as parallel machines or flexible flow or even job shops. We are also interested in extending the sampling method to the energy-aware scheduling problem studied by Rocholl et al. (2020). Moreover, the present implementation does not make use of any parallelization technique. Population-based heuristics often can benefit quite significantly from such techniques, and also the evaluation of samples can be carried out using parallel threads. Therefore, we expect that there is still a huge potential for reducing the computing times and eventually enabling the robust GGA to also cope with large-sized problem instances using an acceptable amount of computing time.

ACKNOWLEDGMENTS

This research was partially supported by the E2BOA project, grant agreement No 005-2011-0128_0111. The authors gratefully acknowledge this financial support.

REFERENCES

Almeder, C., and R. Hartl. 2013. “A Metaheuristic Optimization Approach for a Real-world Stochastic Flexible Flow Shop Problem with Limited Buffer”. *International Journal of Production Economics* 145(1):88-95.

Brown E., and R. Sumichrast. 2003. “Impact of the Replacement Heuristic in a Grouping Genetic Algorithm”. *Computers & Operations Research* 30(11):1575-1593.

Falkenauer, E. 1996. “A Hybrid Grouping Genetic Algorithm for Bin Packing”. *Journal of Heuristics* 2(1):5-30.

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.

Gomez, M. F., V. Borodin, and S. Dauzère-Pérès. 2021. “A Monte Carlo based Method to Maximize the Service Level on the Makespan in the Stochastic Flexible Job-shop Scheduling Problem”. In *Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. August 23rd-27th, Lyon, France, 2072-2077.

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.

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(11): 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.

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(9):1345–1381.

Shahnaghi, K., K. Akbari, S. J. Sadjadi, and M. Heydari. 2016a. "A Scenario-based Robust Optimization Approach for Batch Processing Scheduling". In *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 230(12):2286–2295.

Shahnaghi, K., H. Shahmoradi-Moghadam, A. Noroozi, and H. Mokhtari. 2016b. "A Robust Modelling and Optimisation Framework for a Batch Processing Flow Shop Production System in the Presence of Uncertainties". *International Journal of Computer Integrated Manufacturing* 29(1):92-106.

Sörensen, K. 2001. "Tabu Searching for Robust Solutions". In *Proceedings of the 4th Metaheuristics International Conference*, July 16th-20th, Porto, Portugal, 707-712.

Sevaux, M., and K. Sörensen. 2004. "A Genetic Algorithm for Robust Schedules in a One-machine Environment with Ready Times and Due Dates". *4OR Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 2(2): 129-147.

Sevaux, M., and K. Sörensen. 2009. "A Practical Approach for Robust and Flexible Vehicle Routing using Metaheuristics and Monte Carlo Sampling". *Journal of Mathematical Modelling and Algorithms* 8(4):387-407.

Sobeyko, O., and L. Mönch. 2011. "A Comparison of Heuristics to Solve a Single Machine Batching Problem with Unequal Ready Times of the Jobs". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White and M. Fu, 2006-2016. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Yugma, C., S. Dauzère-Pérès, C. Artigues, A. Derreumaux, and O. Sibille. 2012. "A Batching and Scheduling Algorithm for the Diffusion Area in Semiconductor Manufacturing". *International Journal of Production Research* 50(8): 2118-2132.

Wall, M. 2022. "Galib: A C++ Library of Genetic Algorithms Components." http://lancet.mit.edu/ga/.

AUTHOR BIOGRAPHIES

JENS ROCHOLL is a is a Ph.D. student at the Chair of Enterprise-wide Software Systems, University of Hagen. He received a MS degree in Information Systems from the University of Hagen, Germany. His research interests include scheduling for complex manufacturing systems and metaheuristics. His email address is Jens.Rocholl@fernuni-hagen.de

FAJUN YANG received the B.S. degree in industrial engineering from the Hunan University of Science and Technology, Xiangtan, China, in 2011 and the Ph.D. degree in mechanical engineering from the Guangdong University of Technology, Guangzhou, China, in 2016. From 2015 to 2016, he was a visiting student with the New Jersey Institute of Technology, Newark, NJ, USA. He was Research Fellow with Nanyang Technological University, Singapore. He was also a postdoctorial researcher funded by the Humboldt Foundation at the University of Hagen, Germany. His research interests are cluster tool scheduling. His email adress is fjyang1116@foxmail.com.

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.