*Proceedings of the 2011 Winter Simulation Conference*
*S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.*

# A COMPARISON OF HEURISTICS TO SOLVE A SINGLE MACHINE BATCHING PROBLEM WITH UNEQUAL READY TIMES OF THE JOBS

Oleh Sobeyko
Lars Mönch

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

## ABSTRACT

In this paper, we discuss a scheduling problem for a single batch processing machine that is motivated by problems found in semiconductor manufacturing. The jobs belong to different incompatible families. Only jobs of the same family can be batched together. Unequal ready times of the jobs are assumed. The performance measure of interest is the total weighted tardiness (TWT). We design a hybridized grouping genetic algorithm (HGGA) to tackle this problem. In contrast to related work on genetic algorithms (GAs) for similar problems, the representation used in HGGA is based on a variable number of batches. We compare the HGGA with a variable neighborhood search (VNS) technique with respect to solution quality, computational effectiveness, and impact of the initial solution by using randomly generated problem instances. It turns out that the HGGA performs similar to the VNS scheme with respect to solution quality. At the same time, HGGA is slightly more robust with respect to the quality of the initial solution.

## 1 INTRODUCTION

Scheduling problems for batching machines are important in semiconductor manufacturing, because the processing times of the operations on these machines are large compared to other machines in a wafer fab (Mönch et al. 2011). A batch is a set of jobs that are processed at the same time on the same machine. Recently, this class of scheduling problems has attracted many researchers. Because of the computational complexity of batching problems, heuristics are proposed. GAs are often used as a metaheuristic for parallel batch machine scheduling problems in semiconductor manufacturing. But with the rare exception of (Chiang, Cheng and Fu 2010), the GAs are only used to assign batches to individual machines and then to sequence them. The batch formation decisions are made by dispatching rule based heuristics.

In contrast to previous papers that use GAs to tackle this problem, the GA proposed by Chiang, Cheng and Fu (2010) determines the content of batches and is hybridized by local search. However, the approach has the disadvantage that the number of batches is fixed. GGAs (cf. Falkenauer 1998) can be used quite naturally to tackle this type of problems because each batch can be seen as a group of jobs. The number of jobs within each group is bounded from above by the maximum batch size, i.e., the capacity of the batch processing machine. Because GGAs perform well for multiple orders per job scheduling problems (Mönch and Sobeyko 2011), we are interested in the question whether it is possible to apply GGAs in the present situation or not. Note that in contrast to the multiple order per job situation in this paper the number of batches is variable. In addition, we compare the HGGA approach with a VNS approach with respect to solution quality, computational effectiveness, and the impact of the initial solution. The present paper can be seen as a first step to tackle a larger class of batch scheduling problems in semiconductor manufacturing by GGAs.

The paper is organized as follows. The researched problem is described in the next section. Then we continue by describing different heuristic approaches, especially the HGGA. Computational experiments for comparing the different heuristics are presented in Section 4. Finally, we present some conclusions and plans for future work in Section 5.

## 2 PROBLEM AND RELATED LITERATURE

We consider a single batch processing machine. The maximum batch size is $B$, i.e. at most $B$jobs can be used to form a batch. When a batch is started on a machine no interruption is allowed. Totally, $n$ jobs have to be processed on the batch processing machine. Each job has a due date $d_j$ and a weight $w_j$ that is used to model the importance of the job, and finally a ready time $r_j$. The jobs belong to $F$ incompatible families. Let us denote by $s(j) \in \{1,\ldots,F\}$ the family of job $j$. All jobs of the incompatible job family $f$ have the same processing time $p_f$. Only jobs of the same incompatible family can be used to form a batch. Using the $\alpha|\beta|\gamma$ notation from scheduling theory (cf. Graham et al. 1979) the problem can be represented as follows:

$$1|\text{batch,incomp},r_j|TWT\ , \tag{1}$$

where we denote by TWT the performance measure total weighted tardiness. This measure is defined as $TWT := \sum_{j=1}^{n} w_j \left(C_j - d_j\right)^+$, where we denote by $C_j$ the completion time of job $j$. In addition, we use $x^+ := max(x,0)$ for abbreviation. Problem (1) is NP-hard, because the NP-hard scheduling problem $1||TWT$ can be reduced to it. Note that it might be desirable to form non-full batches, because of the ready times and the *TWT* measure. A schedule consists of a sequence of batches with corresponding starting times. However, in the case where all jobs are ready at time zero, it is shown by Mehta and Uzsoy (1998) that all batches should be full except maybe for the last batch of each family.

Problems of this type have been studied extensively in recent years. We refer to the survey by Mathirajan and Sivakumar (2006). The single machine scheduling problem $1|\text{batch,incomp}|TT$ was studied by Mehta and Uzsoy (1998). Here, we denote by *TT* the total tardiness, i.e., the case where we have $w_j = 1$ for all jobs. This research was later extended to the problem $1|\text{batch,incomp}|TWT$ by Perez et al. (2005). Different metaheuristic approaches including VNS and ant colony optimization are proposed for $\text{P}|\text{batch,incomp}|TWT$ by Almeder and Mönch (2011), where we denote by *P* identical parallel machines. Two genetic algorithms are proposed for the parallel machine scheduling problem $\text{P}|\text{batch,incomp},r_j|TWT$ by Mönch et al. (2005). A VNS approach and a decomposition heuristic based on mixed integer programming is presented by Klemmt et al. (2009) for the problem $\text{R}|\text{batch,incomp},M_j,r_j,B_i|TWT$, where we denote by $R$ unrelated parallel machines, by $M_j$ machine dedications, and by $B_i$ the maximum batch size for machine *i*. Simulated annealing approaches are discussed by Yugma et al. (2008) for a scheduling problem in the diffusion area of a wafer fab that includes batching machines with incompatible job families. Kurz and Mason (2008) propose iterative exchanges procedures for a scheduling problem similar to the present problem.

Recently, a memetic algorithm is proposed for $\text{P}|\text{batch,incomp},r_j|TWT$ by Chiang, Cheng, and Fu (2010). As discussed in Section 1, the resultant GA is used to form batches. However, the number of batches is predetermined for each problem instance before running the GA and cannot be changed by the GA. In this paper, we propose a GGA that is group-centric and allows also for splitting groups, i.e., batches. So far, GGAs are mainly applied to bin packing (Falkenauer 1996) and vehicle routing problems and only rarely for machine scheduling problems.

## 3 HEURISTIC SOLUTION APPROACHES

In this section, we start by recalling a decomposition heuristic that is used for providing initial solutions and reference solutions. Then, we describe the HGGA in detail. For the sake of completeness, we briefly discuss a VNS scheme that is also used for comparison.

### 3.1 Time Window Decomposition Heuristic

We summarize a list scheduling approach for problem (1) based on a time window decomposition (TWD) scheme because initial solutions are determined using this heuristic. The following four steps are performed:

1. When the single machine becomes available at time $t$, only jobs with ready times smaller than $t+\Delta t$ are considered. We use the set $\tilde{J}_f(\mathrm{t}):=\{j \mid \mathrm{r}_\mathrm{j} \le t+\Delta t, s(j)=f\}$ for each family instead of the set of all jobs. We sequence all jobs within $\tilde{J}_f(t)$ with respect to the ATC dispatching rule in non-increasing order. This rule is given by the index:
$$I_j(t):=\frac{w_j}{p_j}exp\left(-\frac{\left(d_j - p_j + (r_j - t)\right)^+}{\kappa\bar{p}}\right), \tag{2}$$
where we denote by $\kappa$ a scaling parameter and by $\bar{p}$ the average processing time of the remaining jobs.
2. Then, we consider only the first *thresh* jobs from the sorted list to form batches. The resulting set is called $\hat{J}_f(t)$.
3. Consider all batches formed by jobs of the set $\hat{J}_f(t)$. Each of these potential batches is assessed using the BATC-II batching rule (Mönch et al. 2005). The corresponding index to assess a batch $b$ of family $f$ is given by
$$I_b(t):=\sum_{\mathrm{j}\in\mathrm{b}}\frac{w_j}{p_f}exp\left(-\frac{\left(d_j - p_j - t + (r_b - t)^+\right)^+}{\kappa\bar{p}}\right)\frac{|b|}{B}, \tag{3}$$
where we denote by $|b|$ the number of jobs in $b$. Here, we use the ready time of batch $b$, defined as $r_b := max(r_j \mid j\in b)$.
4. The batch with the largest BATC-II index is chosen among the different families.
5. The time $t' \le t+\Delta t$ of the next event is calculated, i.e., a machine becomes available or a new job has arrived. Go to Step 1 when there are still jobs not sequenced, otherwise Stop.

In this research, we apply the BATC-II batching rule with an appropriate value of the look-ahead parameter $\kappa$ for sequencing the batches. We select the parameter from the interval $[0.5, 5.0]$ with step size 0.5 and choose the value that corresponds to the smallest TWT value.

### 3.2 HGGA

#### 3.2.1 Representation and Initialization Scheme

In order to determine a feasible schedule jobs have to be assigned to batches and these batches must be sequenced. This particularly implies that appropriate groups of jobs have to be formed. Therefore, it makes sense in a natural way to apply GGA type algorithms when dealing with the problem of batch formation. The used representation is a set of groups. Each group has at most $B$ jobs of the same family. A GGA is a population based metaheuristic. The different individuals from the population are called genomes. A GGA starts from an initial population that can be generated either randomly or a certain part of the population can be obtained as a result of applying heuristics. In this research, we use the TWD heuristic.

### 3.2.2 Genetic Operators

We proceed with describing the main genetic operators of the GGA. The most important one is the crossover operator. It combines parts of the parent genomes in order to obtain child genomes that hopefully correspond to feasible schedules of higher quality. The crossover has to be designed in such a way that child genomes inherit as much valuable information from their parents as possible. For GGAs, the crossover operator has a specific form as it has to deal with groups of jobs rather than with individual jobs. Originally such a crossover operator is proposed by Falkenauer (1996). In this paper, we modify this operator and take into account the incompatible job families. The crossover algorithm consists of the following steps:

1. Select randomly two crossing points, delimiting the crossing section in each parent.
2. Replace the crossing section of parent $G_1$ with the crossing section of parent $G_2$.
3. Eliminate all jobs now occurring twice from the batches where they are contained in the first parent. Thus, some batches originated from the first parent have to be altered.
4. Insert the missing jobs into the child genome and adopt the resulting batches with respect to the hard constraints and the TWT objective.
5. Repeat Step 2 through Step 4 with changed roles of the parents to obtain the second offspring.

The resulting crossover procedure is shown in Figure 1. Ten jobs are involved in this example. The child genome consists of four batches.

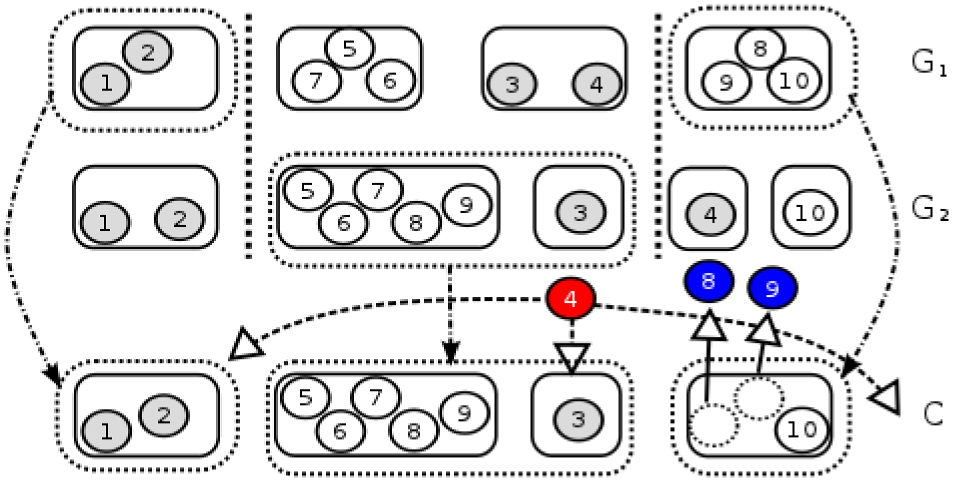

Figure 1: Grouping crossover

As described in Brown and Sumichrast (2003), the reinsertion strategy applied in Step 4 of the crossover algorithm may has a great influence on the performance of the GGA. Here, we tested several reinsertion possibilities and found that one of the best performing strategies is to assign some jobs to the batch only when it does not increase the ready time $r_b$ of the entire batch $b$. In the opposite case, one job with large ready time can cause some delay of the jobs that are already in the batch. Prior to the reinsertion procedure we sort the jobs in descending order according to their weights $w_j$.

A mutation is used to avoid premature convergence of the GA towards local optima. We use a swap type mutation operator, i.e., two batches containing jobs of the same family are selected randomly in some genomes. In each of these batches we select randomly one job and then we exchange the jobs. Another possibility is exchanging more than two jobs in the selected batches.

In order to preserve the most promising parts of the genomes, which includes the batches themselves and their sequence, an inversion genetic operator (Falkenaur 1996 and 1998) is applied. This technique allows a problem-specific sequencing of the formed groups. In this research, we apply the BATC-II batching rule with an appropriate value of the look-ahead parameter $\kappa$ for sequencing the groups. The approach for choosing $\kappa$ is described in Subsection 3.1. A new sequence is accepted only if its TWT value is improved.

### 3.2.3 Local Search Scheme

It is well-known that GAs often do not perform well, unless they are hybridized with local search schemes. Several local search strategies are developed that improve the quality of the genomes in each iteration. The first strategy is a pair-wise job exchange between batches of the same family. The main idea of this procedure is to exchange two jobs only if an improvement of the TWT value is guarantied. We label the batches that are obtained from the genome by their positions in the sequence. We select two different batches $b_i$ and $b_j$ of the same family $f$ and try to consider all possible job exchanges between them as shown in Figure 2.

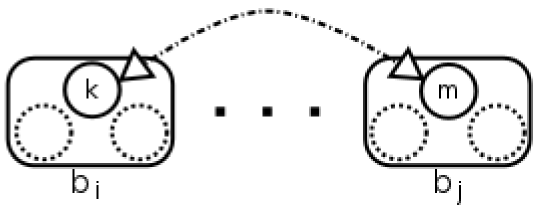

Figure 2: Pair-wise exchange of jobs between batches of the same family

Let $n_{bi}$ and $n_{bj}$ be the number of jobs in batches $b_i$ and $b_j$, respectively. Then the number of possible assignments of the jobs to the two batches is $\begin{pmatrix} n_{bi}+n_{bj} \\ n_{bi} \end{pmatrix}$. This value can be large. We derive some properties that ensure that swaps of jobs across batches lead to TWT reductions without recalculating the TWT value completely. We denote by $C_{bi}$, $C_{bj}$ and $C'_{bi}$, $C'_{bj}$ the completion time of batch $b_i$ and $b_j$ before and after the pair-wise job exchange, respectively. The corresponding ready times of the batches are denoted by $r_{bi}$, $r_{bj}$ and $r'_{bi}$, $r'_{bj}$. The weighted tardiness caused by the batches before and after the swap is denoted by $WT_{bi}$, $WT_{bj}$ and $WT'_{bi}$, $WT'_{bj}$. Here, we define $WT_b := \sum_{k \in b} w_k T_k$ for a batch $b$. When the two conditions

$$C'_{bi} = max\left\{C_{bi-1}, r'_{bi}\right\} + p_f \le max\left\{C_{bi}, r_{bi+1}\right\}, \tag{4}$$

$$C'_{bj} = max\left\{C_{bj-1}, r'_{bj}\right\} + p_f \le max\left\{C_{bj}, r_{bj+1}\right\} \tag{5}$$

hold, then it is easy to see that the entire TWT value associated with the genome is improved, if the condition

$$WT_{bi}^{'} + WT_{bj}^{'} < WT_{bi} + WT_{bj} \tag{6}$$

is fulfilled. The completion times of batches starting from the batch with index $\min\{i, j\}$ have to be updated after an exchange.

Shifting single jobs between batches of the same family is another local search strategy where jobs are removed one by one from their batches and assigned to a different batch. We consider the situation that jobs are removed from batch $b_j$ and assigned to $b_i$, where $i < j$, as depicted in Figure 3.

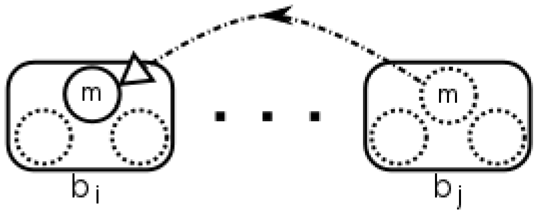

Figure 3 : Job shifting between batches

We see that shifting one job between the batches improves the current value of the objective function if the inequality

$$C_{bi}^{'} = max\{C_{bi-1}, r_i^{'}\} + p_f \le max\{C_{bi}, r_{bi+1}\} \tag{7}$$

implies that

$$WT_{bi}^{'} + WT_{bj}^{'} < WT_{bi} + WT_{bj} \tag{8}$$

is fulfilled. We try to shift all jobs from batch $b_j$. If the reassignment is successful, the completion times of the batches have to be updated starting from batch $b_i$.

Finally, batch splitting is a local search strategy that allows for avoiding situations when jobs with very different ready times form the same batch. Thus, situations when the entire batch is noticeably delayed because it contains jobs with large ready times may be prevented by assigning the jobs that can be processed later to a separate batch. We show the splitting of batch $b_i$ into two batches $b_i^{'}$ and $b_i^{''}$ in Figure 4.

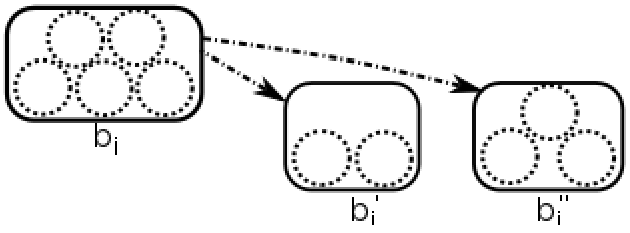

Figure 4: Batch splitting

Such a splitting guarantees the improvement of the TWT value of the corresponding genome if the condition

$$C_{bi}^{''} = max\{r_{bi}^{''}, max\{C_{bi-1}, r_{bi}^{'}\} + p_f\} + p_f < max\{C_{bi}, r_{bi+1}\} \tag{9}$$

implies

$$WT_{bi}^{'} + WT_{bi}^{''} < WT_{bi}.$$

We apply local search within each iteration of HGGA for each new genome directly before calculating its fitness. The local search strategies are executed in the batch splitting $\rightarrow$ pair-wise exchange $\rightarrow$ batch splitting$\rightarrow$shifting$\rightarrow$batch splitting order. Although the local search scheme increases the computational time of the algorithm, it helps to improve the convergence of the GGA significantly.

## 3.3 VNS

The VNS scheme used is similar to the scheme presented by Klemmt et al. (2009) for a slightly more general situation. We consider four different neighborhood structures. All the structures work on the final solution representation, i.e., a sequence of batches. The first neighborhood structure moves single entire batches to a randomly selected position. The second one does the same for sequences of consecutive batches. The third and fourth neighborhood structure swap single batches or sequences of consecutive batches, respectively. The neighborhood structures are applied in this order. The local search approach within VNS is similar to the local search approach used in HGGA to ensure comparability of HGGA and VNS.

## 4 COMPUTATIONAL EXPERIMENTS

This section describes the design of experiments used and computational results which allow us to compare the performance and the computational potential of the heuristics.

### 4.1 Design of Experiments

In order to assess the performance of the different heuristics we randomly generate problem instances according to the design found in (Mönch et al. 2005). The design used is summarized in Table 1.

Table 1: Design of experiments

| Factor | Level | Count |
|---|---|---|
| Number of jobs per family | 60, 80, 100 | 3 |
| Batch size $B$ | 4, 8 | 2 |
| Number of families $F$ | 3 | 1 |
| Family processing time $p_f$ | 2 with probability 0.2<br>4 with probability 0.2<br>10 with probability 0.3<br>16 with probability 0.2<br>20 with probability 0.1 | 1 |
| Job weights $w_j$ | $w_j \sim U(0,1)$ | 1 |
| Ready times of jobs $r_j$ | $r_j \sim U\left(0, \alpha / B \sum p_j\right), \alpha \in \{0.25, 0.5, 0.75\}$ | 3 |
| Due date of jobs $d_j$ | $d_j - r_j \sim U\left(0, \beta / B \sum p_j\right),\ \beta \in \{0.25, 0.5, 0.75\}$ | 3 |
| Total parameter combinations | | 54 |

The parameters $\alpha$ and $\beta$ are used as scaling parameters to control the distribution of ready times and due dates. Larger values of $\alpha$ lead to more spread out release dates. On the other hand, higher values of $\beta$ result in more loose due dates. We consider three different job families. Jobs are assigned to batches and processed on a single batch processing machine with four or eight jobs as maximum batch size. Totally, 54 problem instances are considered.

We implement the HGGA as a steady-state GA with overlapping populations. The parameters of HGGA are presented in Table 2. These parameters are determined based on extensive experimentation using a trial and error strategy. Note the we use a prescribed maximum computing time of two or five minutes, respectively to allow for a fair comparison of the different algorithms.

For TWD we use $\Delta t := \bar{p}/4$, where we denote by $\bar{p}$ the average processing time of the jobs of the problem instance, and $thresh = 15$. All the algorithms are implemented in the C++ programming language. HGGA is coded using the object-oriented framework GALib. A PC with Intel Core i5 2.66 GHz processor and 4 GB of main memory under the OS openSUSE 11.4 Linux is used to carry out the computational experiments. For each problem instance three independent runs with different seed values are performed and the average results over the runs are used for analysis.

Table 2: Parameters of the GGA

| **Parameter** | **Value** |
|---|---|
| Population size | 70 |
| Crossover probability | 0.8 |
| Mutation probability | 0.1 |
| Replacement probability | 0.6 |
| Computing time per problem instance | 2-5 min |

We are interested in comparing the performance of HGGA, VNS, and TWD with respect to solution quality, time needed for computation, and impact of the quality of the initial solution on the overall solution quality.

## 4.2 Results

The obtained computational results are presented in Table 3. Instead of comparing all problem instances individually, the instances were grouped according to levels of factors. For example, the results for $B = 4$ are for all problem instances with $B = 4$ while all other factors have been varied at their different levels. We present the TWT values of the two metaheuristics relative to the TWT values obtained by TWD.

Table 3: Computational results

| **Factor Levels** | **HGGA (init)** | **VNS (in-it)** | **HGGA (no init)** | **VNS (no init)** | **HGGA (no init)** | **VNS (no init)** | **TWD** |
|---|---|---|---|---|---|---|---|
| B = 4 | 0.90 | **0.88** | **0.94** | 0.95 | 0.92 | **0.90** | 1.00 |
| B = 8 | **0.89** | 0.91 | **0.92** | 1.02 | **0.90** | 0.96 | 1.00 |
| n = 180 | **0.90** | 0.91 | **0.92** | 0.98 | **0.91** | 0.92 | 1.00 |
| n = 240 | 0.85 | 0.85 | **0.88** | 0.93 | **0.86** | 0.89 | 1.00 |
| n = 300 | 0.91 | **0.90** | **0.95** | 0.99 | 0.93 | 0.93 | 1.00 |
| $\alpha$ = 0.25 | 0.91 | **0.90** | 0.93 | 0.93 | 0.92 | **0.91** | 1.00 |
| $\alpha$ = 0.5 | **0.87** | 0.88 | **0.91** | 1.00 | **0.89** | 0.94 | 1.00 |
| $\alpha$ = 0.75 | 0.82 | 0.82 | **1.05** | 1.36 | **0.89** | 1.06 | 1.00 |
| $\beta$ = 0.25 | 0.92 | 0.92 | **0.94** | 0.98 | **0.93** | 0.94 | 1.00 |
| $\beta$ = 0.5 | 0.82 | 0.82 | **0.87** | 0.93 | **0.84** | 0.88 | 1.00 |
| $\beta$ = 0.75 | 0.84 | **0.76** | 1.19 | **1.11** | 1.10 | **0.93** | 1.00 |
| **Overall** | 0.89 | 0.89 | **0.93** | 0.97 | **0.91** | 0.92 | 1.00 |
| **Time per problem in-stance (min)** | 2 | 2 | 2 | 2 | 5 | 5 | - |

It appears that both the HGGA and the VNS scheme may have a different performance depending on the initial solutions provided to the algorithms. Therefore, we consider two cases: one with random initial solutions denoted by HGGA (no init) and VNS (no init) and one with initializing the algorithms with rather high-quality solutions obtained from TWD. In case of HGGA, we run TWD with three different

$\kappa \in (0.0, 5.0]$ values to determine different high-quality solutions. Best results for HGGA and VNS with and without appropriate initialization are marked as bold in Table 3.

The results in Table 3 show that both the HGGA and the VNS scheme clearly outperform the TWD scheme in most cases. However, if there is a random initialization of the algorithms they require a rather large amount of time to find high-quality solutions. Sometimes the algorithms are outperformed by the TWD. On the other hand, if the HGGA and the VNS scheme are initialized with solutions obtained by TWD, their performance increases significantly. Overall, both metaheuristics are able to obtain about ten percent improvement over the results found by TWD. Thus, it makes sense to compare the performance of the HGGA and the VNS scheme.

Both the HGGA and the VNS scheme are able to outperform TWD with a reasonable amount of computing time. The performance of the algorithms can differ depending on the factors. Our computational experiments show that the results are quite comparable in most of the situations. However, the HGGA slightly outperforms the VNS scheme in case of a large maximal batch size, i.e. $B = 8$, and a small or moderate number of jobs. If the number of jobs is large then VNS is preferable, since it is not population-based and is therefore faster than the HGGA. It is interesting to note that when one minute computing time per problem instance is used (not shown here), the VNS scheme clearly outperforms HGGA with respect to solution quality.

We are also interested in the effect of the combinations of the level values for $\alpha$ and $\beta$, since they reflect requests of the customers. The corresponding computational results are shown in Table 4.

Table 4: Effect of ready time and due date setting

| **Factor Levels** | **HGGA (init)** | **VNS (init)** | **HGGA (no init)** | **VNS (no init)** | **HGGA (no init)** | **VNS (no init)** | **TWD** |
|---|---|---|---|---|---|---|---|
| $\alpha = 0.25$, $\beta = 0.25$ | 0.94 | **0.93** | **0.94** | 0.95 | 0.93 | 0.93 | 1.00 |
| $\alpha = 0.25$, $\beta = 0.5$ | 0.86 | **0.85** | 0.87 | **0.86** | 0.87 | **0.86** | 1.00 |
| $\alpha = 0.25$, $\beta = 0.75$ | 0.85 | **0.77** | 1.12 | **0.96** | 1.06 | **0.88** | 1.00 |
| $\alpha = 0.5$, $\beta = 0.25$ | **0.91** | 0.92 | **0.93** | 0.99 | **0.91** | 0.95 | 1.00 |
| $\alpha = 0.5$, $\beta = 0.5$ | 0.74 | 0.74 | **0.81** | 0.98 | **0.74** | 0.87 | 1.00 |
| $\alpha = 0.5$, $\beta = 0.75$ | 0.84 | **0.78** | **3.01** | 3.77 | 2.25 | **2.02** | 1.00 |
| $\alpha = 0.75$, $\beta = 0.25$ | **0.85** | 0.86 | **1.01** | 1.20 | **0.90** | 0.99 | 1.00 |
| $\alpha = 0.75$, $\beta = 0.5$ | 0.63 | 0.63 | **1.37** | 2.37 | **0.94** | 1.58 | 1.00 |
| $\alpha = 0.75$, $\beta = 0.75$ | 0.33 | **0.30** | **0.46** | 2.60 | **0.25** | 1.14 | 1.00 |
| **Time per problem instance (min)** | 2 | 2 | 2 | 2 | 5 | 5 | - |

An analysis of the corresponding computational results shows that the HGGA performs slightly better if there is a relatively large range of job ready times and the range of the due dates is moderate. Otherwise, the VNS seems to be able to find better solutions. It is interesting to see that for wide spread ready time and loose due dates HGGA and VNS improve the TWD results up to 70%. This result is caused by the fact that the performance of ATC type heuristics is low in this situation. The overall performance analysis shows that there is little difference between the HGGA and the VNS scheme. According to our results, the HGGA performs similar to the VNS scheme.

## 5 CONCLUSIONS AND FUTUTURE RESEARCH

In this paper, we provide a comparison of several heuristics for solving a single batch machine scheduling problem. Iterative metaheuristics such as GGA and VNS hybridized with local search techniques are able to find high-quality solutions relatively quickly. The performance of both metaheuristics can be significantly improved if initial solutions found by other heuristics, for example TWD, are provided. Applying local search techniques within both the GGA and the VNS increase the performance of the algorithms dramatically. This particularly implies that incorporating problem specific knowledge is of great importance for these metaheuristics. An overall comparison of the computational results shows that both metaheuristics perform quite well and may compete with each other. Although the HGGA has much potential for solving batching problems, the VNS algorithm seems to be able to find high-quality solutions faster than the HGGA. This can be explained by the fact that VNS operates only on one single solution at each point of time, while HGGA maintains a population of solutions. Therefore, it is advantageous to apply VNS in case high-quality solutions have to be found relatively fast. The HGGA can be used if several minutes of computational time per problem instance are available.

As a next logical step the HGGA can be extended to the case of machine environments with parallel machines. An appropriate genome representation has be developed to tackle this problem, especially in case of unrelated parallel machines with dedications. It is interesting to compare the performances of GGA and VNS type heuristics to find out whether both of them are able to find solutions of similar quality and how time-consuming the algorithms are.

## REFERENCES

Almeder, C., and L. Mönch. 2011. “Metaheuristics for scheduling jobs with incompatible families on parallel batching machines.” *Journal of the Operational Research Society*. Accepted for publication.

Brown E., and R. Sumichrast. 2003. Impact of the replacement heuristic in a grouping genetic algorithm. *Computers & Operations Research,* 30:1575-1593.

Chiang, T.-C., H.-C. Cheng, and L.-C. Fu. 2010. “A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families.” *Computers & Operations Research,* 37:2257-2269.

Falkenauer, E. 1996. "A hybrid grouping genetic algorithm for bin packing." *Journal of Heuristics,* 2:5-30.

Falkenauer, E. 1998. *Genetic Algorithms and Grouping Problems*. Wiley.

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.

Klemmt, A., G. Weigert, C. Almeder, and L. Mönch. 2009. “A comparison of MIP-based decomposition techniques and VNS approaches for batch scheduling problems.” In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1686-1694. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kurz, M., and S. J. Mason. 2008. “Minimizing total weighted tardiness on a batch-processing machine with incompatible job families and job ready times.” *International Journal of Production Research,* 46(1): 131-151.

Mönch, L., H. Balasubramanian, J. W. Fowler, and M. E. Pfund. 2005. “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.* Accepted for publication.

Mathirajan, M., and A. Sivakumar. 2006. “A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor.” *International Journal of Advanced Manufacturing. Technology* 29: 990-1001.

Mehta, S. V., and R. Uzsoy. 1998. “Minimizing total tardiness on a batch processing machine with incompatible job families.” *IIE Transactions,* 30: 165-178.

Perez, I., J. Fowler, and W. Carlyle. 2005. “Minimizing total weighted tardiness on a single batch processing machine with incompatible job families.” *Computers & Operations Research,* 32: 327-341.

Sobeyko, O., and L. Mönch. 2011. “Grouping genetic algorithms for solving single machine multiple orders per job scheduling problems.” Working paper, University of Hagen.

Yugma, C., S. Dauzère-Pérès, A. Derreumaux, and O. Sibille 2008. “A batch optimization software for diffusion area scheduling in semiconductor manufacturing.” *Advanced Semiconductor Manufacturing Conferenc 2008 (ASMC 2008).*

## AUTHOR BIOGRAPHIES

**OLEH SOBEYKO** is a PhD student and research and teaching assistant in the Department of Mathematics and Computer Science at the University of Hagen, Germany. He received a master’s degree in applied mathematics from the National Ivan Franko University of Lviv, Ukraine. His current research interests are in applied optimization applications in manufacturing, logistics and service operations, and in multi-agent systems. His email address is <Oleh.Sobeyko@fernuni-hagen.de>.

**LARS MÖNCH** is Professor in the Department of Mathematics and Computer Science at the University of Hagen, Germany. He received a master’s degree in applied mathematics and a Ph.D. in the same subject from the University of Göttingen, Germany. His current research interests are in simulation-based production control of semiconductor wafer fabrication facilities, applied optimization and artificial intelligence applications in manufacturing, logistics, and service operations. He is a member of GI (German Chapter of the ACM), GOR (German Operations Research Society), SCS, INFORMS, and IIE. His email address is <Lars.Moench@fernuni-hagen.de>.