Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

Decomposition heuristics for parallel-machine multiple orders per job scheduling problems with a common due date

Jens Rocholl & Lars Mönch

To cite this article: Jens Rocholl & Lars Mönch (2019): Decomposition heuristics for parallel-machine multiple orders per job scheduling problems with a common due date, Journal of the Operational Research Society, DOI: 10.1080/01605682.2019.1640589

To link to this article: https://doi.org/10.1080/01605682.2019.1640589

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
https://doi.org/10.1080/01605682.2019.1640589

ORIGINAL ARTICLE

Decomposition heuristics for parallel-machine multiple orders per job scheduling problems with a common due date

Jens Rocholl and Lars Mönch

Department of Mathematics and Computer Science, University of Hagen, Hagen, Germany

ABSTRACT

Scheduling problems for identical parallel machines with earliness-tardiness objective are studied that are motivated by manufacturing processes in 300-mm wafer fabs. Wafers are transferred by front opening unified pods (FOUPs) in such fabs. Only a limited number of FOUPs is allowed since a large number of FOUPs results in a highly congested automated material handling system. A FOUP can contain a group of orders. A nonrestrictive common due date is assumed for all orders. Only orders of the same family can be grouped together in a FOUP. The lot and the item processing mode are differentiated in this article. Mixed integer linear programming (MILP) models are provided for both modes. It is shown that the two scheduling problems are NP-hard. Simple decomposition heuristics based on list scheduling and bin packing procedures are proposed. Biased random-key genetic algorithm (BRKGA)-based decomposition schemes are designed for the two scheduling problems. The BRKGAs are hybridised with the simple heuristics and an integer programming-based job formation approach in the lot processing mode. Results of computational experiments based on randomly generated problem instances are analysed and discussed for both scheduling problems. The results show that the proposed heuristics perform well with respect to solution quality and computing time. The BRKGA-type approaches clearly outperform the other heuristics.

ARTICLE HISTORY
Received 30 May 2018
Accepted 29 June 2019

KEYWORDS
Parallel machine scheduling; common due date; multiple orders per job; decomposition; semiconductor manufacturing

1. Introduction

The manufacturing processes in semiconductor wafer fabrication facilities (wafer fabs) belong to the most complex processes that exist today in manufacturing. Integrated circuits (ICs or chips) are processed in wafer fabs starting from a thin disc made from silicon or gallium arsenide, a so-called wafer. Several thousands of chips can be produced on the surface of a single wafer. The diameter of wafers is 200 or 300 mm (Mönch, Fowler, & Mason, 2013). Although the transition to 450 mm wafers is discussed controversially, it is expected that wafers with such a diameter will play a crucial role in next-generation wafer fabs (Semi, 2018).

ICs are built up by at most 40 layers onto a raw wafer. Lots of wafers, we call them jobs to go conform with the literature on scheduling theory, form the moving entities in wafer fabs. Several process conditions are characteristic for wafer fabs that make them different from a conventional job shop setting. These process conditions are:

1. A single process flow might contain up to 800 process steps.
2. The processing takes place on several hundreds of machines that form machine groups. A machine group contains machines with the same or very similar functionality. There are very expensive machines in wafer fabs.
3. Reentrant process flows are common in wafer fabs, that is, each job might visit the same machine group up to 40 times for the most advanced technologies.
4. Single wafer processing, lot processing, and batch processing occur. Here, a (parallel) batch, for short a p-batch, is a group of jobs that are processed together, that is, in parallel, at the same time on the same machine (Mönch, Fowler, Dauzère-Pérès, Mason, & Rose, 2011).
5. An automated material handling system (AMHS) can be found in most wafer fabs (Agrawal & Heragu, 2006).

Dispatching approaches are widely applied in wafer fabs. Dispatching rules support production control activities. However, due to the increasing degree of automation in wafer fabs, scheduling approaches become more popular. At the same time, scheduling approaches have become more competitive as a result of the dramatic increase in computer efficiency. Scheduling approaches can be found in modern wafer fabs on the machine group

CONTACT Lars Mönch lars.moench@fernuni-hagen.de Department of Mathematics and Computer Science, University of Hagen, Hagen 58097, Germany

Supplemental data for this article is available online at https://doi.org/10.1080/01605682.2019.1640589.

level and on the level of work areas, that is, a collection of several machine groups (Mönch et al., 2013). Known successful scheduling applications in wafer fabs are described in the survey article by Mönch et al. (2011). One representative example of a real-world implementation can be found in Yugma, Dauzère-Pérès, Artigues, Derreumaux, and Sibille (2012).

A front opening unified pod (FOUP) is a standard carrier for up to 25 300-mm wafers in an inert, nitrogen atmosphere (Agrawal & Heragu, 2006). The number of FOUPs in a wafer fab is fairly small since an AMHS may become congested if many FOUPS are in place. In addition, FOUPs are expensive. The scheduling problem considered in this article is motivated by the fact that less 300 mm wafers are required to fulfil an order of a customer than in 200 mm wafer fabs due to the combination of increased area per 300 mm-wafer and of decreased chip line width. Because of this, it is often desirable to group different orders into a single FOUP. Jobs are the resulting entities. All orders that form a job have to belong to the same family due to the different nature of the family-dependent manufacturing processes. After jobs are formed, they must be scheduled on the machines. These scheduling problems are called multiple orders per job (MOJ) problems since a grouping of several orders into a single job takes place (Mönch et al., 2011).

Identical parallel-machine MOJ scheduling problem is considered in this article since the machinery of wafer fabs consists of machine groups. Both the lot processing and the item processing modes of the machines are studied. The processing time of an individual job is just the wafer processing time in the former mode. In the latter mode, the processing time of a job is the sum of the identical processing times of the wafers belonging to the orders of the job. A nonrestrictive late common due date, often also called "unrestrictively late" common due date, is assumed. Base wafers found in many wafer fabs, especially in foundries which manufacture products for other semiconductor companies (Mönch et al., 2013) serve as motivation for this assumption. Base wafers are preprocessed wafers where a certain number of layers are already processed. They are held on stock for customer requests due to contractual obligations. The due dates of almost all wafers belonging to the customer orders are the same and typically not tight since there exists some time flexibility for a foundry to fulfil the contracts.

In this article, the sum of the absolute deviations of the order completion times from a nonrestrictive late common due date is the performance measure. This so-called earliness/tardiness (E/T) objective is essential for just-in-time manufacturing. A large tardiness value leads to customer dissatisfaction, while a large earliness value indicates inventory holding costs. To the best of our knowledge, this MOJ scheduling problem is not studied yet. It is an extension of the single-machine MOJ scheduling problem with a common due date and lot processing mode that is investigated by Rocholl and Mönch (2018). A preliminary version of the parallel-machine scheduling problem with lot processing mode is presented in the extended abstract by Rocholl and Mönch (2017). However, this article includes a rather complete analysis of the scheduling problems for the lot and the item processing mode together with an algorithmic description and a rigor computational assessment of the proposed heuristics.

The article is organised as follows. The MOJ scheduling problems are introduced and analysed in the following section. This includes a mixed integer linear programming (MILP) formulation and the analysis of the complexity status. We also discuss important related work. Structural properties of optimal schedules are proven in Section 3. Decomposition approaches for the two processing modes are proposed in Section 4. Computational results are discussed and analysed in Section 5. Section 6 discusses conclusions and directions for future research.

2. Problem analysis and related work

2.1. Problem setting

The set of all orders is denoted by O. Each order o has a size s_o given as the number of items (wafers) belonging to that order. Overall, we have N orders from L different families. Only orders of the same family can be used to form a job. It is assumed that $N_l, l = 1, \ldots, L$ orders are in family l. The family of order o is $f(o)$. It is not allowed to split orders into suborders. The ready time of all orders is zero. Wafers from orders of family l have an identical processing time ρ_l for both processing modes. Since all wafers of a job are processed together in the lot processing mode, the processing time p_j of a job j is determined by the family of the orders that are grouped into j. Hence it holds $p_j = \rho_l$ when l is the family of the orders that belong to job j. Wafers are processed sequentially in the item processing mode. Therefore, the processing time of a job j that consists of orders o with $l = f(o)$ is $p_j = \rho_l \sum_{o \in j} s_o$. Moreover, job availability is considered requiring that an order is not available until the entire job to which it belongs is completed. Job availability refers to the situation that the completion time of all orders of a job is the same. Moreover, F FOUPs are available. They have an identical capacity of K items. At most F jobs are possible. The set of all jobs is J. Jobs are labelled by $j = 1, \ldots, |J|$ where

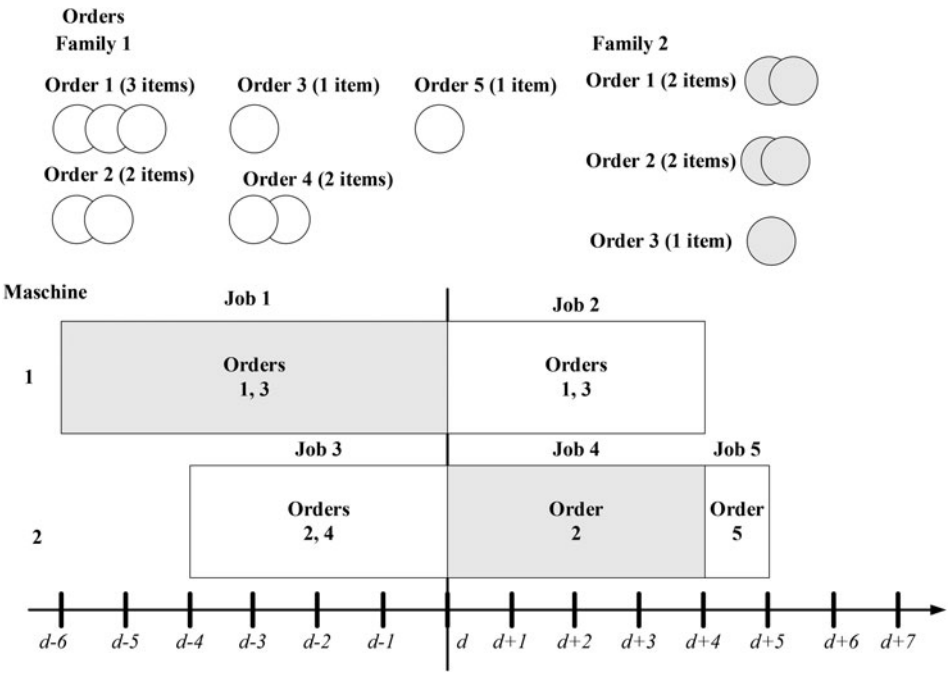

Figure 1. Feasible schedule for problem (5).

$|J| \leq F$ holds. The completion time of job j is e_j, while C_o is the completion time of order o. The family of the orders that belong to job j is $\phi(j)$.

We assume that m identical parallel machines exist. The processing of jobs on a machine cannot be interrupted after the job processing has been started. The same due date $d_o \equiv d$ is assumed for all orders. This common due date is nonrestrictive, that is, it holds

$$\sum_{l=1}^{L} N_l \rho_l \leq d \tag{1}$$

in the lot processing mode and

$$\sum_{l=1}^{L} \sum_{\{o|f(o)=l\}} \rho_l s_o \leq d \tag{2}$$

in the item processing mode. The minimisation of the E/T value of the orders is the goal where the earliness of order o is $E_o := (d - C_o)^+$ and the tardiness is $T_o := (C_o - d)^+$. Here, we set $x^+ = \max(x, 0)$. The E/T objective function is

$$E/T := \sum_{o=1}^{N} (E_o + T_o) = \sum_{o=1}^{N} |C_o - d|. \tag{3}$$

Applying the $\alpha|\beta|\gamma$ notation (Graham, Lawler, Lenstra, & Rinnooy Kan, 1979), the two scheduling problems are represented by

$$P|\text{moj(lot)}, \text{incompatible}, d_o \equiv d|E/T \tag{4}$$

and

$$P|\text{moj(item)}, \text{incompatible}, d_o \equiv d|E/T \tag{5}$$

where identical parallel machines are indicated by P, moj(lot) and moj(item) refer to the lot and item processing mode and "incompatible" to the order families. Moreover, $d_o \equiv d$ is the common due date.

The following three decisions must be made when instances of the two scheduling problems are solved:

1. *Job formation*: The grouping of orders into jobs is the major decision to be made.
2. *Job assignment*: The already formed jobs must be assigned to individual machines from the set of parallel machines.
3. *Job sequencing*: The jobs have to be sequenced on each of the machines.

The three decisions are interrelated since, for example, the job processing time is determined by the job content in the item processing mode. However, in the lot processing mode, the job assignment decisions influence the job sequencing decisions. Hence, a simultaneous treatment of the three decisions is desirable but the problem then becomes very complex. Therefore, we will design several decomposition approaches in this article.

Next, we consider a small-sized example for problem (5). Two families and two machines in parallel are given. The processing times per item for the first and second family are 1 and 2. We have $K = 4$ items. Different colours are used in Figure 1 to differentiate between the orders of the different families. Five jobs are formed, assigned to the two machines, and sequenced there. A feasible schedule, for example, is shown in Figure 1.

The E/T value of the schedule is $E/T = 2 \cdot 0 + 2 \cdot 4 + 2 \cdot 0 + 1 \cdot 4 + 1 \cdot 5 = 17$. It is important that the common due date is large enough. The

concrete value is not interesting since the order completion times are determined relative to the $d$value.

2.2. Discussion of related work

We start by discussing single- and parallel-machine scheduling with a common due date and E/T objective, especially for batch processing machines and for a MOJ setting. Kanet (1981) shows in his pioneering article that $1|d_j \equiv d|E/T$ can be tackled using a fairly simple algorithm that is able to provide an optimal schedule at low computational costs. Generalisations of the Kanet algorithm to parallel-machine environments are proposed by Emmons (1987) and by Hall (1986). The scheduling problem from Kanet (1981) is generalised by Hall and Posner (1991). A weighted E/T objective function is studied resulting in a NP-hard problem. Structural properties for certain classes of optimal schedules are derived. Moreover, constructive heuristics are proposed. We refer to Gordon, Proth, and Chu (2002), Lauff and Werner (2004), and Jozefowska (2007) for surveys of scheduling research with a common due date and E/T objective function to obtain a more complete picture of research for this class of scheduling problems.

Next, batch scheduling problems are discussed. The processing time of a p-batch is given as the longest processing time of the jobs, that is, the scheduling entities that are part of this batch. We refer to Mathirajan and Sivakumar (2006) and Mönch et al. (2011) for surveys of p-batching scheduling problems in wafer fabs. Since batching decisions are grouping decisions by nature, scheduling problems for batching machines that include a common due date are somehow similar to the problem (4) in this article. However, we discuss only articles with a common due date and the E/T objective function due to space limitations. The problem $1|p-\text{batch}, d_j \equiv d, \Delta|E/T$ is studied by Mönch et al. (2006). The quantity Δ is the maximum allowed tardiness. A GA that incorporates structural properties for a certain class of optimal schedules is proposed. Zhao, Hu, and Li (2006) consider $1|p-\text{batch}, CDW|WE/T$, where CDWrefers to a common due window and WE/T to a variant of the E/T performance measure where job-independent weights for early and tardy jobs exist. Mönch and Unbehaun (2007) propose GAs for $P|p-\text{batch}, d_j \equiv d|E/T$. The heuristics from this article are extended by Li, Chen, Xu, and Li (2015) to a single-machine setting where nonidentical sizes of the jobs are allowed. This is similar to MOJ problems where orders with nonidentical sizes can exist. Moreover, exact and heuristic algorithms for the problem discussed by Li et al. (2015) are recently designed by Parsa, Karimi, and Moattar Husseini (2017). Several mathematical programming approaches for batching problems with a WE/T measure are proposed by Ogun and Alabas-Uslu (2018). The main difference between the scheduling problems for p-batching and problem (4) of this article is that in the former the number of batches is not a constraint while in the latter one we have a maximum number of FOUPS. Therefore, methods for p-batching cannot be directly used for problem (4) of this article.

Problem (5) shows some similarity to s-batching problems with batch availability, also called serial batching. In this setting, a batch is a set of jobs that are scheduled contiguously, that is, in a serial manner on a machine and share often a setup (Potts & Kovalyov, 2000). The processing time of a batch in s-batching is the sum of the processing times of the jobs that belong to the s-batch. A maximum batch size might be assumed based on the capacity of the machines or due to manufacturing reasons. Examples for the former setting are given by Pei et al. (2017a, 2017b) and Pei et al., (2019), while the scheduling problem discussed by Mönch (2002) belongs to the latter situation. But again, the major difference between s-batching scheduling problems and problem (5) is that the number of batches is arbitrary in s-batching, while we have a maximum number of FOUPS in the MOJ item processing case. Moreover, family-dependent setups are not considered in the MOJ setting. Therefore, scheduling algorithms for problem (5) of this article cannot rely much on previous research on s-batching. This is especially true since we are only aware of the articles by Azizoglu and Webster (1997) and Suriyaarachchi and Wirth (2004) where a single-machine s-batching problem with a weighted E/T measure and a nonrestrictive common due date are studied. Job availability is assumed. Branch & bound algorithms and several fast heuristics and a GA are proposed for this problem.

Next, we discuss MOJ scheduling research for machine environments that contain machine groups, that is, parallel machines. Column generation techniques are applied to the problem $P|r_o, \text{moj(lot)}|\text{TWC}$ by Jampani and Mason (2008), where the total weighted completion time objective is considered. Moreover, unequal ready times r_o of the orders are assumed. Jia and Mason (2009) design and analyse heuristic algorithms for $P|r_o, s_{kl}, \text{moj(lot)}|\text{TWC}$, where s_{kl} indicates sequence-dependent setup times.

Only a few articles consider MOJ problems in a flow shop and job shop setting. Exact and heuristic approaches are analysed by Laub, Fowler, and Keha (2007) for the two-stage flow job scheduling problem $F2|\text{moj(item)}|C_{\max}$. Here, $C_{\max}$ refers to the

makespan. Sarin, Wang, and Cheng (2012) study a generalised version of this problem. A hybrid heuristic that combines ant colony optimisation and constraint programming is used by Jampani, Pohl, Mason, and Mönch (2010) to solve the reentrant flexible job shop problem $FJ|\text{moj(lot)}, r_o, \text{recrc}|\text{TWC}$. However, in this article, we assume a common due date and the E/T measure which is not considered in these previous MOJ articles for different non-single machine environments and regular performance measures. The non-restrictive common due date and the E/T measure allow for heuristics that exploit special structure properties of certain classes of optimal schedules.

The references most pertinent to this article are the article by Rocholl and Mönch (2018), where the problem $1|\text{moj(lot)}, \text{incompatible}, d_o \equiv d|E/T$ is studied and several GA-based heuristics are proposed and the extended abstract by Rocholl and Mönch (2017), where the present authors sketch very briefly heuristics for $P|\text{moj(lot)}, \text{incompatible}, d_o \equiv d|E/T$. In this article, we extend these two references towards parallel machines and the item processing mode. Both extensions are reasonable since item processing and machine groups are common in wafer fab settings (Mönch et al., 2013). Compared with the extended abstract by Rocholl and Mönch (2017), the proposed heuristics for the lot processing mode will be described in much more detail and much more results of computational experiments will be shown and analysed in this article. Overall, based on our discussion of related literature, it seems that parallel-machine MOJ scheduling problems with a nonrestrictive common due date are not fully studied so far.

2.3. MILP formulations and complexity status

A MILP formulation for the problems (4) and (5) is proposed. Without loss of generality, we assume that we have a fixed sequence of the jobs. This sequence is used to label the jobs. If we have $i<j$ for two jobs and the jobs are processed on the same machine then job i always starts not later than job j. Next, indices and sets are presented:

$1 \le j, h \le F$: job indices
$o = 1, ..., N$: order indices
$l = 1, ..., L$: family indices
$k = 1, ..., m$: machine indices.

The following decision variables are used:

$$x_{oj} := \begin{cases} 1, & \text{if order } o \text{ belongs to job } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jl} := \begin{cases} 1, \text{if job } j \text{ is formed by orders of family } l \\ 0, \text{otherwise} \end{cases}$$

$$z_{jk} := \begin{cases} 1, & \text{job } j \text{ is assigned to machine } k \\ 0, & \text{otherwise} \end{cases}$$

e_j : completion time of job j
C_o : completion time of order o
E_o : earliness of order o
T_o : tardiness of order o.

The following parameters are used by the model:
s_o : size of order o (in number of items)
d : common due date
K : capacity of a FOUP (in number of items)
ρ_l : processing time of an item of family l
G : very large number.

Next, the model itself is formulated as follows:

$$\min \sum_{o=1}^{N} (E_o + T_o) \tag{6}$$

subject to

$$\sum_{j=1}^{F} x_{oj} = 1, \ o = 1, ..., N, \tag{7}$$

$$\sum_{o=1}^{N} s_o x_{oj} \le K, \ j = 1, ..., F, \tag{8}$$

$$\sum_{l=1}^{L} y_{jl} = 1, \ j = 1, ..., F \tag{9}$$

$$x_{oj} - y_{jf(o)} \le 0, o = 1, ..., N, \ j = 1, ..., F \tag{10}$$

$$\sum_{k=1}^{m} z_{jk} = 1, \ j = 1, ..., F \tag{11}$$

$$e_j + \rho_l y_{hl} - G(2 - z_{jk} - z_{hk}) \le e_h, \ j, h \in \{1, ..., F\}, \ j < h, k = 1, ..., m, l = 1, ..., L$$

$$e_j + \rho_l \sum_{o=1}^{N} s_o x_{oh} - G(2 - z_{jk} - z_{hk}) - G(1 - y_{hl}) \le e_h, o = 1, ..., N, j = 1, ..., F \tag{12a}$$

$$h = 1, ..., F, j < h, k = 1, ..., m, \ l = 1, ..., L \tag{12b}$$

$$e_j - G(1 - x_{oj}) \le C_o, o = 1, ..., N, \ j = 1, ..., F \tag{13}$$

$$C_o \le e_j + G(1 - x_{oj}), o = 1, ..., N, \ j = 1, ..., F \tag{14}$$

$$C_o - d \le T_o, \ o = 1, ..., N \tag{15}$$

$$d - C_o \le E_o, \ o = 1, ..., N, \tag{16}$$

$$x_{oj}, \ y_{jl}, z_{jk} \in \{0, 1\}, 0 \le C_o, e_j, E_o, T_o, o = 1, ..., N, \ j = 1, ..., F, l = 1, ..., L, k = 1, ..., m. \tag{17}$$

The objective function (6) that is to be minimised is the sum of the E/T values of the individual orders. Each order is assigned to exactly one job by constraints (7). The capacity of the jobs is modelled

by the constraints (8). Constraints (9) ensure that only a single family is assigned to a job. Constraints (10) make sure that the orders of a job have an identical family. Moreover, constraint set (11) is responsible that each job is processed on exactly one machine. In lot processing mode, constraints (12a) relate the completion time of jobs to the sequence of the jobs on each machine and the corresponding processing times of the jobs, while constraints (12b) must be applied for this purpose in the item processing mode. The completion time of the orders of each job is computed based on the constraint sets (13) and (14). The individual tardiness and earliness of the orders is linearized by the constraint sets (15) and (16). The decision variable domains are respected by the constraint set (17). Note that the MILP (6)–(17) might schedule empty jobs. It is easy to see that they always can be placed after nonempty jobs in an optimal solution.

We will later use the MILP formulation (6)–(17) to check the correct implementation of the different heuristics by computing optimal schedules for instances of small size. The MILP cannot be used to solve medium- or large-sized instances to optimality within a reasonable amount of computing time. This behaviour is caused by the complexity status of problem (4) and (5), respectively. We know from (Rocholl & Mönch, 2018) that even the single-machine, single-family version of problem (4), that is, the problem $1|\text{moj(lot)}, d_o \equiv d|E/T$, is NP-hard. In this article, we prove an analogous result for the problem $1|\text{moj(item)}, d_o \equiv d|E/T$ using the fact that the 3-partition problem is strongly NP-hard (Garey & Johnson, 1979). We start by recalling **3-partition**:

> It is assumed that $3M$ integers $a_j, j = 1, \ldots, 3M$ exist with $B/4 < a_j < B/2, j = 1, \ldots, 3M$ and $\sum_{j=1}^{3m} a_j = MB$ for a positive integer B. Are there M pairwise disjoint sets $A_i \subseteq \{1, \ldots, 3m\}$ each consisting of three elements such that $\sum_{j \in A_i} a_j = B$ for $i = 1, \ldots, M$?

Next, we state the following theorem.

Theorem 1: The scheduling problem $1|\text{moj(item)}, d_o \equiv d|E/T$ is NP-hard in the strong sense.

Proof: The proof is based on a specific single-family instance of problem (5) that is provided by the choices $N := 3M$, $F := M$, $s_j := a_j$, $K := B$, and $\rho_1 := 1$. The threshold of the E/T value is $Y := 3K \sum_{j=1}^{F}((\lceil F/2 \rceil - j)^+ + (j - \lceil F/2 \rceil)^+)$. The decision version of the scheduling problem $1|\text{moj(item)}, C_o \equiv d|E/T$ asks whether there is a feasible schedule S such that $E/T(S) \leq Y$ holds or not. We will show that a feasible schedule for this instance with E/T value not larger than Y exists if and only if 3-partition has a solution for the given input data.

For the first direction, we assume that 3-partion has a solution. This leads to $A_i, i = 1, \ldots, M$ with $\sum_{j \in A_i} s_j = K, i = 1, \ldots, M$. Each order is assigned to exactly one of the A_i sets, since the A_i are disjoint and $\sum_{i=1}^{3M} s_i = MK$ holds. The E/T value of the corresponding schedule is then equal to Y since all jobs contain K wafers.

For the reverse direction, we start from a feasible schedule for the instance with E/T value not larger than the threshold value. Let us assume that a job has more than three orders. This contradicts $K/4 < s_o, o = 1, \ldots, N$ and $\sum_{j \in o} s_o \leq K$ for a job j. If at most two orders belong to a job j then $\sum_{o \in j} s_o < K$ follows from the conditions $s_o < K/2$, $o = 1, \ldots, N$. We obtain $\sum_{o=1}^{N} s_o < FK$ which contradicts $\sum_{o=1}^{N} s_o = FK$. Therefore, it is demonstrated that we have F jobs where exactly three orders belong to each job and that each job has K items. This defines a solution of 3-partition. ■

Since problem (5) contains the scheduling problem $1|\text{moj(item)}, d_o \equiv d|E/T$ as a special case, problem (5) is NP-hard too. As a consequence, we will concentrate on efficient heuristics in the rest of the article. The design of such heuristics can be facilitated when structural properties of certain classes of optimal solutions are known that can be exploited for the heuristics.

Next, we assume that assignment and sequencing decisions for all jobs are already made and the family is known for each job. Metaheuristics can be applied for this purpose (see Section 4.3). An integer programming (IP) model for the remaining subproblem of order-to-job assignment can be then formulated for the lot processing mode since in this situation the family of a job determines its processing time. Here, we set $\Delta_j := |e_j - d|$ for abbreviation. The resulting model can be formulated as follows:

$$\min \sum_{j=1}^{F} \sum_{o=1}^{N} \Delta_j x_{oj} \tag{18}$$

subject to

$$\sum_{j=1}^{F} x_{oj} = 1, o = 1, \ldots, N \tag{19}$$

$$\sum_{o=1}^{N} s_o x_{oj} \leq K, j = 1, \ldots, F \tag{20}$$

$$\big(f(o) - \phi(j)\big) x_{oj} = 0, o = 1, \ldots, N, j = 1, \ldots, F \tag{21}$$

$$x_{oj} \in \{0, 1\}, o = 1, \ldots, N, j = 1, \ldots, F. \tag{22}$$

The E/T value of the schedule is given by the objective function value (18). The constraint set (19) ensures that each order belongs to a single job. The constraints (20) model the capacity restriction of the jobs. The equations (21) make sure that an order can only belong to a job if the job and the order have the same family. The constraints (22) ensure that the decision variables are binary.

The IP (18)–(22) is similar to the generalised assignment problem. Note that even this problem is known to be NP-hard it can be solved to optimality in a reasonable amount of time for fairly large-sized problem instances. This fact will be exploited when we design metaheuristics in Section 4. The IP (18)–(22) is abbreviated by MOJAP.

3. Structural properties of optimal schedules

3.1. Mode-independent properties

The job availability leads to the same completion time of all orders of a job and therefore to a situation where the E/T value can be computed as weighted sum over a single order representative of each job. The weight of a job corresponds to the number of orders included in this job. Therefore, structural properties of optimal solutions of the scheduling problem $1|d_j \equiv d|WE/T$ carry over from Hall and Posner (1991) to the present MOJ setting.

The first property holds for any optimal schedule for instances of problem (4) and (5), respectively.

Property 1: There is no idle time between two arbitrary consecutive jobs in any optimal schedule for instances of problem (4) or (5).

A proof of this property can be obtained using the same arguments as in Rocholl and Mönch (2018).

The second property shows that optimal schedules without any straddling jobs exist. A straddling job starts before and finishes after d. This property is important since we know from it and from Property 1 that on each machine with assigned jobs a job has to be completed at d and that another tardy job if such a job exists at all must start at d.

Property 2: An optimal schedule exists for instances of the problems (4) or (5) such that a job is completed at d on each machine where jobs are scheduled.

This property can be proved using the same arguments as in Rocholl and Mönch (2018).

The following third property is important since it allows for characterising the job sequence of single machines. $\hat{E}_k$ is the set of all early jobs and the job that completes processing at d on machine k, while $\hat{T}_k$ is the tardy job set on machine k. The quantity $|J_j|$ is the number of orders included in job j.

Property 3: An optimal schedule exists for instances of the problems (4) or (5) such that the jobs from $\hat{E}_k \neq \emptyset$ are sorted with respect to $|J_j|/p_j$ in non-decreasing order. Furthermore, the jobs from $\hat{T}_k \neq \emptyset$ are sorted with respect to $|J_j|/p_j$ in non-increasing order.

This property can be shown using the same arguments as in Rocholl and Mönch, (2018) for the single-machine case. Property 3 results in so-called V-shaped schedules (Hall & Posner, 1991). This means that the jobs that are on each machine before the common due date are sequenced based on the weighted longest processing time (WLPT) rule in non-decreasing order, while jobs after the common due date are sequenced using the weighed shortest processing time rule (WSPT) in non-increasing order.

Property 4 deals with the assignment of jobs to $\hat{E}_k$ and $\hat{T}_k$ in arbitrary optimal schedules.

Property 4: The sum of the processing times of the jobs in any set $\hat{T}_k \neq \emptyset$ is not larger than the sum of the processing times of the jobs in the set $\hat{E}_k \neq \emptyset$ in any optimal schedule for instances of problem (4) or (5).

This property can be shown in a MOJ context by recalling the arguments from Rocholl and Mönch (2018) for jobs on any nonempty machine.

The next property is related to a balanced workload of the different machines. A similar result can be found in a slightly different context in (Kim, Kim, & Lee, 2012).

Property 5: Two different machines k and u are considered. Let $j^{(l)}$ be the last scheduled job in the set $\hat{T}_k \neq \emptyset$ of any optimal schedule S for instances of problem (4) and (5). The corresponding job at the first position in $\hat{E}_k \neq \emptyset$ is $j^{(f)}$. We then have (a) $\sum_{j\in\hat{T}_k} p_j \leq \sum_{h\in\hat{T}_u} p_h + p_{j^{(l)}}$ and (b) $\sum_{j\in\hat{E}_k} p_j \leq \sum_{h\in\hat{E}_u} p_h + p_{j^{(f)}}$.

Proof: The existence of an optimal schedule S with $\sum_{j\in\hat{T}_k} p_j > \sum_{h\in\hat{T}_u} p_h + p_{j^{(l)}}$ is assumed. We remove job $j^{(l)}$ from k and insert it on the last position in $\hat{T}_u$. The change in the E/T value of the new schedule compared with S is $|J_{j^{(l)}}|(\sum_{h\in\hat{T}_u} p_h + p_{j^{(l)}} - \sum_{j\in\hat{T}_k} p_j) < 0$, which contradicts the optimality of S. The proof for part (b) works similar. ■

3.2. Properties for the lot processing mode

We know from Mason, Qu, Kutanoglu, and Fowler (2004) that it is beneficial to use a small number of jobs in the lot processing mode, while the maximum number of possible jobs is preferred in the item processing mode since job availability is assumed. Therefore, we show next several properties for problem (4) that strive for an appropriate formation of jobs for the lot processing mode.

Property 6: Let j and h be two different jobs with $\varphi(j) = \varphi(h)$ in an arbitrary optimal schedule for instances of problem (4). Then the largest size of any order that belongs to job j is unequal to the sum of the order sizes of at least two orders of job h when $\Delta_j < \Delta_h$ holds.

Proof: Assume that an optimal schedule S exists with jobs j and h such that $\varphi(j) = \varphi(h)$ and $\Delta_j < \Delta_h$. Furthermore, let $o$be an order from j and the two orders p, q belong to job h such that $s_o =$

$s_p + s_q$. Exchanging the three orders such that o belongs to h and p, q to j leads to a change in the E/T value of $2\Delta_j - 2\Delta_h + \Delta_h - \Delta_j = \Delta_j - \Delta_h < 0$ due to $\varphi(j) = \varphi(h)$ which contradicts the optimality of S. ■

Note that generalisations of Property 6 are possible such that the sum of the order sizes from n_1 orders that belong to job j is in any optimal schedule always unequal to the sum of the order sizes of n_2 orders of job h as long we have $\Delta_j < \Delta_h$ and $n_1 < n_2$.

Property 7 deals with the capacity utilisation of jobs for the lot processing mode.

Property 7: Let j and h be different jobs with $\varphi(j) = \varphi(h)$ in an arbitrary optimal schedule for instances of problem (4). Then the available capacity in job j is smaller than the smallest size of any order of job h if $\Delta_j < \Delta_h$ holds.

Proof: The existence of an optimal schedule S with jobs j and h such that $\varphi(j) = \varphi(h)$ and $\Delta_j < \Delta_h$ is assumed. Moreover, let o be an order from h such that o can be inserted into j without exceeding the capacity of j. The change in the E/T value is then $\Delta_j - \Delta_h < 0$ which is a contradiction to the optimality of S. ■

Property 6 justifies that it may be favorable for problem (4) to form jobs of orders grouped by their size. Jobs with a large number of small-sized orders are desirable because the E/T value of these orders is small if such jobs can be scheduled close to d. Property 7 indicates that higher utilised jobs are favorable.

3.3. Properties for the item processing mode

Next, we consider the item processing mode. According to Property 2, there is a job with completion time d on each machine if we assume that $F \geq m$ holds. The E/T value for these m jobs is zero. It is reasonable to assign as much as possible orders to these m jobs if these jobs are the first ones on each machine. In this situation, it is reasonable focusing on the remaining tardy jobs. It is possible to state the following property for tardy jobs.

Property 8: If there is a tardy job with at least two orders in an optimal schedule of an instance of problem (5) then all jobs are used, that is, we have $|J| = F$.

Proof: Assume that an optimal schedule S exists with tardy jobs j and h, where we have $|J_j| = 0$, that is, this job is not used so far and $|J_h| \geq 2$. The contribution of any two orders o, p from h to the E/T value of S is $\Delta(S, o, p) = 2\rho(s_o + s_p)$, where we set $\rho = \rho_{f(o)} = \rho_{f(p)}$. It is possible to place job j right after job h without increasing the tardiness of any order. Let S' be a schedule where order p is moved from h to j. The contribution of the orders o, p to the E/T value of the schedule S' is $\Delta(S', o, p) = \rho s_o + \rho(s_o + s_p) < \Delta(S, o, p)$ which is a contradiction to the optimality of S. ■

While Property 8 deals with tardy orders, the next property shows that it is under several conditions beneficial to assign a larger number of early orders to the same job.

Property 9: In any optimal schedule for instances of problem (5), any two orders of the same family that are sequenced in two adjacent jobs in $\hat{E}_k$ can be assigned to the job that is closer to the common due dates as long as the capacity constraint of this job is not violated.

Proof: Assume that we have an optimal schedule S where orders p and q with $f(p) = f(q)$ and $C_p < C_q \leq d$ are assigned to two adjacent jobs. Inserting order p into the job where order q belongs to leads to a schedule S' where C_p is increased to the value of C_q while the remaining orders are unaffected by the exchange. The earliness of S' is reduced by the amount $C_q - C_p > 0$ compared with S which contradicts the optimality of S. ■

4. Decomposition approaches

4.1. Reference heuristics

The simple heuristics first proposed in this subsection are designed to solve sequentially the subproblems of job formation, assignment, and sequencing. The orders are sorted according to the smallest size (SS) and the largest size (LS) rule. The index $I(o) := s_o$ is used by the SS rule to sort the orders in non-decreasing order, whereas the LS rule applies $I(o)$ in opposite order. The orders are then assigned to jobs based on order-to-job-assignment rules. We discuss first-fit (FF) strategies from bin packing proposed by Mason et al. (2004) and two new rules. Note that these strategies do not necessarily guarantee that all orders can be assigned to jobs, that is, infeasibilities might be the result in some situations when the given number of FOUPs is small. A fixed job sequence is considered without loss of generality, that is, jobs are labelled by an index. In addition, we assume that the set of all orders is divided into subsets of orders that belong to the same family. The following strategies are used:

1. **FFD1:** Orders are assigned to the first job of a sorted list. If it is impossible to assign more orders to this job due to missing remaining capacity, the next job is taken. The procedure is repeated until no orders are left.
2. **FFD1b:** This rule is very similar to the FFD1 rule. The only difference is that exactly one order is assigned to each of the empty jobs as soon as the number of remaining orders to be assigned is equal to the number of empty jobs.

3. **FFDn:** This rule performs a single pass over the sorted order list. Each order is assigned to the first job with enough available capacity.
4. **FFBF:** This strategy combines a first-fit and a best-fit strategy to achieve a dense packing of orders resulting in a high capacity utilisation of the jobs. Orders are assigned from a sorted list to a job until it is impossible to include orders from this list in the current job due to the capacity constraint. If the capacity of the job is not fully used, the last order which was assigned is removed from the job and reinserted into the order list. The order list is then sorted in non-increasing order of the order sizes. Starting from the beginning of the order list, the first order whose size is equal or less than the remaining capacity of the job is assigned. The procedure is repeated with the next jobs until all orders are assigned.
5. **FFDAJS:** This strategy aims for reaching a similar capacity utilisation of all jobs. This is accomplished by considering the ratios of the sum of the sizes of all unassigned orders and the number of empty jobs. The next job is filled according to the FFD1 taking into account the expected average job size derived before and the other constraints.
6. **FFDR**: Similar to the FFDAJS strategy, this strategy distributes the orders over the entire set of available jobs. The orders are stored in a sorted list. A single pass through the list assigns the orders to jobs. Whenever an order is assigned to a job or cannot be assigned due to insufficient job capacity, the next job in the given sequence of jobs is considered. When the end of the job sequence is reached the assignment will start over. As there are often more orders than jobs, multiple passes through the sequence of jobs are expected.

The FFBF strategy is proposed because if applied in combination with the SS rule this strategy will lead to jobs that have at the the same time a large number of assigned orders and a high capacity utilisation. Therefore, both Properties 6 and 7 are respected.

Note that the FFD1b, FFDAJS, and FFDR strategies require to know the number of unused jobs per family that are available. Therefore, the number of jobs of family l is initially set according to:

$$F_l := \left\lfloor F \sum_{p \in O_l} s_p / \sum_{o \in O} s_o \right\rfloor, l = 1, \ldots, L \qquad (23)$$

where $O_l := \{o \in O | f(o) = l\}$. When jobs are left after this initial setting step they are equally distributed over the families. For the FFD1, FFDn, and FFBF strategies, a job belonging to a certain family is created if orders require a new job of this particular family.

Next, we describe a list scheduling-type algorithm for assigning the formed jobs to the sets $\hat{E}_k$ and $\hat{T}_k$ of the machines. The early-tardy heuristic (ETH) can be stated as follows:

4.1.1 ETH-based list scheduling

1. The set of unscheduled jobs is considered in a prescribed order that is, for example, the result of the simple order-to-job-assignment rules described above or of metaheuristic approaches. Choose the first job from this sequence. This job is called j^*.
2. Calculate the sum of the processing times of the jobs already assigned to the sets $\hat{E}_k$ and $\hat{T}_k$, respectively, to obtain the early and tardy jobs sets with the smallest processing time. These sets are $\hat{E}_k$ and $\hat{T}_r$, respectively. Note that $k = r$ is possible.
3. Insert job j^* at the first position of the set $\hat{E}_k$ if the sum of the processing time of the jobs from $\hat{E}_k$ is smaller than the sum of the processing times of the jobs from $\hat{T}_r$ and p_{j^*}. Otherwise, insert job j^* at the last position of the set $\hat{T}_r$.
4. Remove job j^* from the set of all unscheduled jobs and repeat Step 1 if unscheduled jobs exist.
5. The early and tardy jobs on all machines are sorted according to the WLPT and WSPT rule, respectively.

Note that the first m jobs are scheduled by the ETH scheme to being completed at d. Thus, Property 2 is respected. Furthermore, the scheme will produce a schedule that fulfils the Properties 3, 4, and 5. Heuristics based on the ETH scheme are abbreviated by ETH(assignment strategy – order sorting rule). For example, the heuristic indicated by the notation ETH(FFBF-SS) first assigns orders to jobs based to the FFBF strategy that is applied to a set of orders sorted by the SS rule, then assigns the formed jobs using the list scheduling approach and sorts the early and tardy job sets of each machine by the WLPT and WSPT rule, respectively. The ETH-type decomposition heuristics will be used for the performance assessment of GA-based approaches for both problem (4) and (5).

Next, we sketch an iterated local search (ILS) procedure as an alternative to the simple heuristics. It is well-known that such type of neighbourhood search approaches often performs well for hard combinatorial optimisation problems (Voß and Woodruff, 2006). An initial solution is constructed by applying the ETH(FFBF-SS) heuristic for problem (4) and the ETH(FFDAJS-LS) heuristic for problem (5), respectively. Local search is

implemented considering pair-wise interchange of the machine assignments and positions of two jobs. Within each step one job is randomly chosen. All possible swap operations for this job are considered, that is, the effect on the E/T value of swapping the job with any job which is either assigned to a different machine or scheduled at the opposite side of the common due date is evaluated. If necessary, the jobs of the affected machines are re-sequenced to comply with Property 3 to obtain a V-shaped schedule before the evaluation. If the E/T value can be reduced, the swap leading to the largest reduction is performed. Each local search phase is limited to a number of λ steps. The resulting solution is then perturbed to obtain a new starting point for consequent iterations of local search. The perturbation procedure operates on the same neighbourhood structure. The machine assignments and positions of two randomly chosen jobs are interchanged. This operation is repeated for τ times per perturbation process. The stopping criterion of the overall procedure is given by a prescribed time limit.

4.2. Random-key genetic algorithm and biased random-key genetic algorithm principles

GAs evolve a set of chromosomes, also called a population, representing solutions of the given optimisation problem over a number of generations. A fitness value that is computed based on the objective function value is assigned to each chromosome. The chromosomes of a new generation are generated from the previous generation by applying selection rules and crossover and mutation operators (compare Michalewicz, 1996).

Random-key genetic algorithm (RKGA) are proposed by Bean (1994) for sequencing problems. The chromosomes are vectors of randomly generated real numbers, so-called random keys. A decoder associates each single chromosome with a solution for which the objective function value can be computed. The random-key vector is sorted to obtain a sequence (Bean, 1994). A randomly chosen initial population of random-key vectors is used to start the GA. The fitness of the chromosomes of the population is then calculated. The population is partitioned into a small set of elite chromosomes and a larger set of non-elite chromosomes. Elite chromosomes have a high fitness. The chromosomes of the elite set of generation g are transferred into the generation $g+1$. An immigration strategy is applied for mutation. Uniform crossover is used to determine the remaining chromosomes of the population. For this, first two chromosomes of the population are randomly chosen. A biased coin is then tossed for each gene to determine the parent contributing

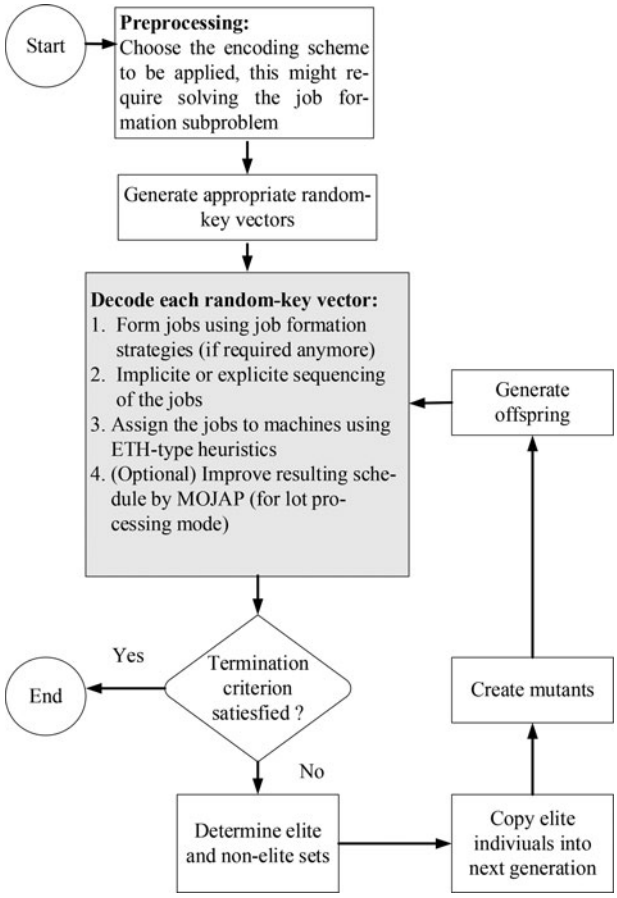

Figure 2. Generic flow of BRKGA-type heuristics for the problems (4) and (5).

to the corresponding allele. Biased RKGAs (BRKGAs) are designed by Gonçalves and Resende (2011) as a RKGA variant where always one of the parents used for mating is selected from the elite set and the second parent is chosen randomly from the non-elite set. Moreover, the first parent is chosen to contribute to the allele during uniform crossover with a probability $\rho_e > 0.5$, that is, more alleles are often taken from the parent belonging to the elite set. Due to $\rho_e > 0.5$ this RKGA variant is called biased. It is reported by Gonçalves and Resende (2011) that BRKGAs outperform RKGAs for many combinatorial optimisation problems.

Note that only the encoding and decoding schemes have to be specified to design a BRKGA for a specific problem. This approach will be applied in the next two subsections to described specific BRKGA-based heuristics for problem (4) and (5), respectively. The generic flow of the heuristics is shown in Figure 2.

4.3. BRKGA-type procedures for the lot processing mode

In this subsection, we will propose two BRKGA-based algorithms for problem (4). Using RKGA-type approaches is motivated by the fact that such

algorithms perform well for the single-machine version of problem (4) that is studied by Rocholl and Mönch (2018).

In the first BRKGA, a chromosome is an array of F random keys from the interval $(0, 1)$ that determine the position of the job in the sequence of all jobs and eventually the family of the job. To reduce the risk of having not enough jobs to cover all the orders of a family a minimum number of jobs per family is set by

$$F_l = \lceil Q_l \rceil, l = 1, ..., L, \tag{24}$$

where $Q_l = \sum_{o \in O_l} s_o / K$ can be considered as an estimate for the number of fully loaded jobs of family l. The family of the remaining jobs is determined by the chromosome as follows. Let $r_h \in (0, 1)$ be the random key associated with job h. We decode the family of job h by $\varphi(h) := \lceil L r_h \rceil$, while the fractional quantities $L r_h - \lfloor L r_h \rfloor$ determine the sequence of the jobs. The empty jobs are then assigned to the sets $\hat{E}_k$ and $\hat{T}_k$ using the ETH approach from Section 4.1. Orders are assigned to the jobs in the schedule using the FFBF-SS strategy for each of the $\hat{E}_k$ and $\hat{T}_k$ sets separately. The FFBF-SS strategy is chosen due to its superior performance (see Section 5.3).

The resulting E/T value can be improved based on the MOJAP IP (18)–(22) from Section 2.3 by changing the order-to-job assignment of the schedule. Since solving a large number of MOJAP IP instances tends to be time-consuming the following strategy motivated by the matheuristic proposed by Mönch and Roob (2018) is applied. The MOJAP IP is only generated and solved when the E/T value associated with the chromosome is smaller than $\gamma E/T(ETH(FFBF - SS))$, where $E/T(ETH(FFBF - SS))$ refers to the E/T value of a schedule obtained by the ETH(FFBF-SS) procedure and γ is a given threshold value that is reduced by a certain factor each time an IP is solved. Starting from an initial value γ_0, the threshold value decreases by multiplying γ with a constant value $\psi < 1$. By reducing the γ value, it is less likely that the IP is generated and solved. Hence, the BRKGA is able to perform enough generations. Moreover, the amount of computing time used by the IP can be controlled by a maximum computing time or by a prescribed MIP gap. The MOJAP IP is applied again to the fittest chromosome after the BRKGA termination condition is fulfilled. This first variant is abbreviated by BRKGA-FR since the job families are determined, that is, ranked, by the GA.

The second BRKGA uses again a job-based random-key representation. We start by forming jobs based on the FFBF-SS strategy from Section 4.1 to determine the length of the random-key vectors. In a second step, the BRKGA provides random-key vectors which are decoded into job sequences. Recall that we know for each job j its family and therefore its processing time $\rho_{\varphi(j)}$. These job sequences are the input for Step 1 of the ETH procedure from Section 4.1. After performing the ETH procedure, the E/T value associated with the corresponding chromosome is computed. If the resulting ETH value is smaller than the current threshold value, the MOJAP IP is applied to improve the E/T value by changing the content of the jobs. The MOJAP IP is again applied to the schedule associated with the best chromosome after the BRKGA is terminated. We refer to this variant as BRKGA-JR due to the fact that the jobs are ranked by the GA.

4.4. BRKGA-type procedures for the item processing mode

The BRKGA-FR and BRKGA-JR procedures are designed to solve problem (4). Both algorithms tend to form jobs with a small amount of unused capacity. They compute schedules that often do not utilise all available jobs. In view of Property (8), we expect that this behaviour leads to poor schedules for problem (5). Moreover, the MOJAP IP cannot be applied for problem (5) since the processing times of the jobs depend on the sizes of the orders that belong to the jobs.

Therefore, three BRKGA-type heuristics are proposed for problem (5). The first one uses an order-based encoding scheme. The chromosomes are random-key vectors of length N. An order sequence can be derived from each chromosome. These sequences are used by the order-to-job-assignment strategies from Section 4.1 to form the jobs. From the six strategies, FFD1b, FFDAJS, and FFDR make use of all available jobs. In the light of Property 8 a fairly good performance of schedules found by these strategies is expected. We know from computational experiments (see Section 5.3) that the FFDAJS strategy often results in slightly better schedules than the FFD1b strategy while the FFDR strategy is clearly outperformed. We therefore apply the FFDAJS strategy for job formation. After job formation, again the ETH approach is applied. The resulting heuristic is called BRKGA-IOR since an order ranking is performed for the item processing mode.

The second heuristic is based on the idea that it is favourable to use all available jobs. Therefore, a random-key vector with F components is used to encode job sequences. The algorithm starts by assigning orders to jobs according to the FFDAJS-LS strategy. The jobs then are sorted using their random-key values. This job sequence serves as input for the ETH approach. This heuristic is

Table 1. Summary of the different BRKGA-type heuristics.

BRKGA	Encoding scheme	Job formation	Job sequencing	Final schedule
FR	Job-based	FFBF-SS	BRKGA	ETH for early and tardy job sets, MOJAP IP
JR	Based on jobs from FFBF-SS	–	BRKGA	ETH, MOJAP IP
IOR	Order-based	FFDAJS	implicit	ETH
IJR	Job-based	FFDAJ-LS	BRKGA	ETH
IOET	Order-based	FFBF-SS for early, FFDAJS-LS for tardy	implicit	ETH for early and tardy job sets

Table 2. DOE for medium- and large-sized problem instances.

Factor	Level	Count
N	30,60,120,240	4
s_o	$\sim DU\left[\nu-\frac{\nu+1}{2},\nu+\frac{\nu+1}{2}\right],\nu\in\{3,5\}$	2
K (in number of items)	$12\beta+1,\ \beta\in\{1,2\}$	2
L	3,10	2
Item processing time distribution	5 with $p=0.2$, 4 with $p=0.2$, 10 with $p=0.3$, 16 with $p=0.2$ 20 with $p=0.1$	1
m	2, 6	2
Number of independent replications	10 per factor combination	10
Total number of problem instances		640

abbreviated by BRKGA-IJR since it is based on job ranking.

The third approach for solving problem (5) is again based on random-key vectors of length N. Let $r_o \in (0,1)$ be the random key associated with order o. Order o is early if $\lceil 2r_o \rceil = 1$ holds, whereas o is tardy when $\lceil 2r_o \rceil = 2$ is fulfilled. Sorting the fractional quantities $2r_o - \lfloor 2r_o \rfloor$ provide the order sequences. Different strategies are pursued for the order-to-job assignment of the two order sets. A compact assignment seems desirable for the early orders in view of Property 9. Thus the FFBF-SS strategy is applied. This strategy is able to produce a dense assignment using a fairly small number of jobs. The jobs that do not contain any order after applying the FFBF-SS strategy are then used to form tardy jobs. The job formation using tardy jobs is carried out by the FFDAJS-LS strategy to comply with Property 8. After the early and tardy jobs are formed, they are assigned to the parallel machines by list scheduling similar to the ETH approach. The early jobs are assigned to the machine with the smallest value for the sum of the processing times of the already assigned early jobs. The tardy jobs are assigned to the machines in an analogous manner. Finally, Step 5 of the ETH procedure is performed to ensure that Property 3 is respected. This heuristic is called BRKGA-IOET to indicate that order ranks are used to form early and tardy order sets. The different BRKGA-type heuristics for the problems (4) and (5) are summarised in Table 1.

5. Computational experiments

5.1. Design of experiments

Randomly generated problem instances depending on N, L, m, and K are considered within the experimentation since we expect that these factors influence the performance of the proposed algorithms. The number of FOUPs in the instances is $F = \lceil N\nu/12\beta \rceil + L$, where the ν values influence the size of the orders, and β is used to set the FOUP capacity. The family-dependent processing times for a single item are instance-specific and follow a prescribed discrete distribution. We always have $N_l := N/L$. Table 2 contains the chosen experimental design (DOE) for 640 medium- and large-sized problem instances. The design reflects important process conditions such as the number of orders in a typical wafer fab (Mönch et al., 2013).

The relative performance of the BRKGA-based heuristics is compared with a simple reference heuristic suitable for the particular problem. We therefore choose the performance ratio (PR) of a heuristic H as

$$PR_{lot} := \mathrm{E/T}(H)/\mathrm{E/T}(ETH(FFBF-SS)), \quad (25)$$

$$PR_{item} := \mathrm{E/T}(H)/\mathrm{E/T}(ETH(FFDAJS-LS)) \quad (26)$$

for problems (4) and (5). The heuristics $ETH(FBFBF-SS)$ and $ETH(FFDAJS-LS)$ are chosen among the strategies from Section 4.1 for comparison since we know from experiments with the 640 instances from Table 2 that they are able to find very often feasible solutions. Furthermore, additional computational experiments are performed for 40 small-sized problem instances according to the DOE in Table 3 to assess the performance of the heuristics based on optimal E/T values obtained from the MILP formulations (6)–(17).

The corresponding PR indices for the problems (4) and (5) are $PR_{opt,lot}(H) = \frac{E/T(H)}{E/T(MILP)}$ and $PR_{opt,item}(H) := \frac{E/T(H)}{E/T(MILP)}$, respectively. For the factor combination $L = 3$, $\nu = 5$, and $\beta = 1$, an optimality proof is impossible within the prescribed maximum

Table 3. DOE for small-sized problem instances.

Factor	Level	Count
N	15	1
s_o	$\sim DU\left[\nu-\frac{\nu+1}{2},\nu+\frac{\nu+1}{2}\right],\nu\in\{3,5\}$	2
K (in number of items)	$12\beta+1,\ \beta\in\{1,2\}$	2
L	3, 5	2
Item processing time distribution	5 with $p=0.2$, 4 with $p=0.2$, 10 with $p=0.3$, 16 with $p=0.2$ 20 with $p=0.1$	1
m	2	1
Number of independent replications	5 per factor combination	5
Total number of problem instances		40

computing time of 10 min per instance using CPLEX 12.1. In this situation, the best E/T value found by the MILP is used to compute the PR indices. Five independent replications are performed for the BRKGA-based heuristics. Average values are reported for all performance measures.

For the BRKGA-IOR, BRKGA-IJR, and BRKGA-IOET, we allow 60 s of computing time per instance for all medium- and large-sized instances and 15 s for small-sized instances. For BRKGA-FR and BRKGA-JR, separate computing time limits are used for the pure BRKGA iterations and for solving single MOJAP IP instances. Table 4 summarises the chosen settings depending on F. We always use the same amount of computing time for the ILS scheme as for the corresponding BRKGA variant.

We refer to the sum of the computing time for the pure BRKGA part and for solving the MOJAP instance after performing the BRKGA as combined time limit.

5.2. Parameterisations of the algorithms and implementation details

We use $\lambda=50$ and $\tau=100$ in the ILS scheme. These settings are determined following a trial and error strategy. The population size is set to 300 for all BRKGA variants. The fraction of the elite set within the entire population is selected as $p_e=0.15$. A fraction of the population to be replaced by mutants is $p_m=0.2$. The probability that an offspring inherits the allele of its elite parent is chosen as $\rho_e=0.6$. To determine these settings some preliminary experimentation with 64 problem instances of problem (4) was conducted. The tested values for the population size are {100, 200, 300}. We also test the settings $p_e\in\{0.05,0.10,0.15\}$, $p_m\in\{0.1,0.2,0.3\}$, and $\rho_e\in\{0.6,0.7,0.8\}$. To reduce the computational burden, only parameter combinations from a Taguchi Orthogonal Array are used for both the BRKGA-FR and the BRKGA-JR. The found parameter settings also perform well for the BRKGA variants of problem (5). The MOJAP IP application described in Section 4.3 is parameterised by choosing $\gamma_0=1.4$ and $\psi=0.99999$ based on trial and error. The C++ programming language is used to code all the algorithms. The BRKGA variants are based on the brkgaAPI framework (compare brkgaAPI, 2018) proposed by Toso and Resende (2011). The MILP and the MOJAP IP instances are solved by the CPLEX 12.1 libraries. The computational experiments are performed on an Intel Core i7-2600 CPU 3.40 GHz PC with 16 GB RAM.

5.3. Computational results

We start by assessing the ETH-type heuristics from Section 4.1 based on the instances from Table 1. The corresponding PR_{lot} and PR_{item} values are depicted in the Figures 3 and 4, respectively.

The FFBF, FFD1, and FFDn strategies strive for using a small number of jobs. Therefore, only these strategies are applied for problem (4). The FFD1b, FFDAJS, and FFDR strategies are considered for problem (5) since only these strategies tend to use all available jobs. The ETH(FFBF-LS) scheme is equivalent to ETH(FFD1-LS) and therefore not considered separately. Applying the SS rule often causes a heuristic to fail at finding a feasible solution when jobs are scarce except for the ETH(FFBF-SS) heuristic. Therefore, the ETH(FFD1-SS) and ETH(FFDn-SS) schemes are not considered anymore in Figure 3. The same is true for the ETH(FFD1b-SS), ETH(FFDAJS-SS), and ETH(FFDR-SS) schemes in Figure 4.

Next, we assess the heuristics based on optimal solutions for problem (4). Instead of individually comparing the 40 instances from Table 3, the results are grouped by factor levels such as L, ν, and β. Each cell contains the average value of five instances. The average $PR_{opt,lot}$ values are summarised in Table 5. The detailed results for the ILS scheme are not shown here due to space limitation, however, we obtain an overall value of $PR_{opt,lot}=1.048$ for the ILS procedure.

The results for problem (5) are presented in the same way. When a heuristic cannot come up with a feasible solution the n/a symbol is used to indicate this. The average $PR_{opt,item}$ values are collected in Table 6. We obtain an overall value of $PR_{opt,item}=1.595$ for the ILS scheme.

Table 4. Computing time limits for problem (4) (in s).

Number of jobs	pure BRKGA part	MOJAP IP within a BRKGA iteration	MOJAP IP after the BRKGA
≤ 15	10	2	5
16 - 29	50	5	10
≥ 30	150	10	30

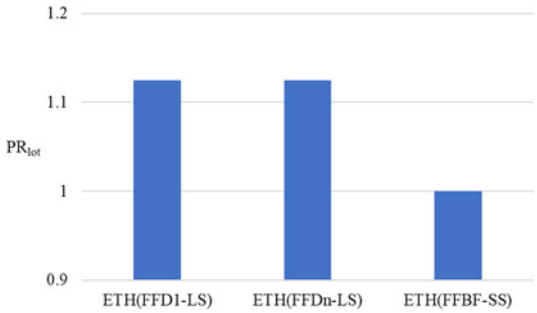

Figure 3. Average PR_{lot} values obtained by simple heuristics (lot processing mode).

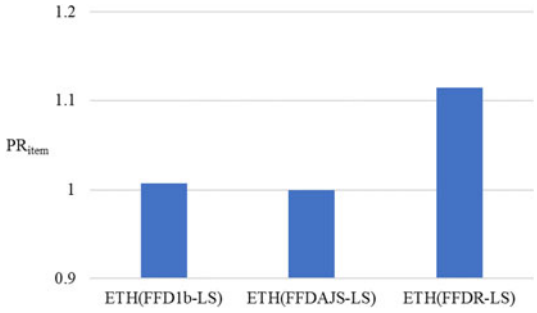

Figure 4. Average PR_{item} values obtained by simple heuristics (item processing mode).

Because the MILP (6)-(17) cannot prove the optimality of the solutions for problem instances with $N > 15$ in a reasonable amount of time, we present the average PR_{lot} values for medium- and large-sized instances of problem (4) in Appendix Tables A1 and A2 (supplementary material). The instances are grouped by the factor levels for N, m, L, ν, and β. Each row contains the average PR_{lot} values of 10 instances. Best values in a given row are marked bold. The combined time limit (see Section 5.1) for the BRKGA-type heuristics is also reported. The overall values for the ILS scheme are $PR_{lot} = 0.922$ and $PR_{lot} = 0.924$ for $N = 30$ and $N = 60$, respectively. The corresponding results for $N = 120$ and $N = 240$ are $PR_{lot} = 0.950$ and $PR_{lot} = 0.969$.

The results of medium- and large-sized instances are presented in the same way in Appendix Tables A3 and A4 (supplementary material). However, the computing times of the BRKGA-type schemes are not reported in contrast to Appendix Tables A1 and A2 (supplementary material) since they are constant at 60 s per instance. The overall values for the ILS scheme are $PR_{item} = 0.841$ and $PR_{item} = 0.862$ for $N = 30$ and $N = 60$, respectively. The corresponding results for $N = 120$ and $N = 240$ are $PR_{item} = 0.925$ and $PR_{item} = 0.965$.

Table 5. Computational results for small-sized instances (lot processing mode).

L	ν	β	ETH(FFBF-SS)	BRKGA-FR	BRKGA-JR
3	3	1	**1.000**	**1.000**	**1.000**
	3	2	**1.000**	**1.000**	**1.000**
	5	1	1.147	**1.000**	**1.000**
	5	2	1.067	**1.000**	**1.000**
5	3	1	1.048	**1.000**	**1.000**
	3	2	1.108	**1.000**	**1.000**
	5	1	1.103	**1.000**	**1.000**
	5	2	1.159	**1.000**	**1.000**
Overall			1.079	**1.000**	**1.000**

5.4. Analysis and discussion of the results

The Figures 3 and 4 confirm that the ETH-type heuristics that are used for comparison with the BRKGA-type approaches, namely the ETH(FFBF-SS) and the ETH(FFDAJS-LS) procedures outperform the remaining simple heuristics.

We see from Table 5 that the BRKGA-FR and the BRKGA-JR are able to determine the optimal solutions for all the small-sized instances. This behaviour is caused by incorporating the MOJAP IP. The solver computes the optimal order-to-job assignment for the chromosomes obtained from the GAs. Both heuristics are based on a job-based representation. There are between 4 and 12 jobs. Thus, the GAs are able to visit a large portion of all job permutations. Consequently, it is likely that optimal job assignments and sequences are computed while the solver provides a high-quality or even optimal order-to-job assignment. In contrast to this, the ETH(FFBF-SS) procedure does not find optimal solutions for all instances. The performance of the ILS scheme is between that of ETH(FFBF-SS) and the BRKGA-type algorithms.

Table 6 contains the corresponding results for problem (5). We see that the BRKGA-IOET outperforms the remaining heuristics. It is able to determine optimal solutions under many experimental conditions. Note that for the factor combination $L = 3$, $\nu = 5$, $\beta = 1$, where CPLEX is not able to find the optimal solutions within the prescribed maximum computing time of 10 min, the E/T values found by BRKGA-IOET within 15 s are on average even slightly smaller. The poor performance of schedules found by the BRKGA-IOR and the BRKGA-IJR demonstrates that these heuristics are not able to address the specific characteristics of small-sized problem instances. The FFDAJS is included to determine job formations that use all available jobs to align with Property 8. However, in any optimal solution there is one job on each machine which is completed at d. It is reasonable to place as many orders as possible in these jobs (see the related discussion in Section 3.3). This may result in situations where some available jobs are unused in the case of small-sized instances, whereas

Table 6. Computational results for small-sized instances (item processing mode).

L	ν	β	ETH(FFDAJS-LS)	BRKGA-IOR	BRKGA-IJR	BRKGA-IOET
3	3	1	2.813	2.122	2.503	**1.000**
	3	2	2.829	2.400	1.903	**1.000**
	5	1	1.475	1.250	1.351	**0.990**
	5	2	n/a	n/a	n/a	**1.091**
5	3	1	1.383	1.348	1.198	**1.053**
	3	2	1.172	1.172	**1.000**	**1.000**
	5	1	1.424	1.242	1.211	**1.023**
	5	2	1.171	1.171	**1.000**	**1.000**
Overall			1.752	1.529	1.452	**1.020**

the heuristics are designed to use as many jobs as possible. The performance of the ILS scheme is again between that of ETH(FFDAJS-LS) and the BRKGA-type algorithms.

The results in the Tables 5 and 6 indicate that the different heuristics are correctly implemented and that the BRKGA-type approaches lead to high-quality solutions for small-sized problem instances.

The results in the Appendix Tables A1 and A2 (supplementary material) demonstrate that the BRKGA-type heuristics provide high-quality solutions. Schedules are obtained for medium-sized instances with $N = 30$ that outperform the schedules found by ETH(FFBF-SS) by around 10% on average. The performance of the ILS scheme is between the performance of the BRKGA-type approaches and that of ETH(FFBF-SS) for $N \leq 120$. The average performance of the BRKGA-FR and the BRKGA-JR schemes seems to be similar. However, the BRKGA-JR determines slightly better solutions when the number of orders increases. A more detailed analysis for the results of the 640 instances based on the Wilcoxon signed-rank test (Conover, 1980) for a 1% significance level confirms that the results of the two heuristics are statistically different, that is, the BRKGA-JR outperforms the BRKGA-FR. The results of this analysis are shown in Table 7 where the '>' sign refers to the situation that the heuristic denoted in the column performs significantly better than the one of the row.

In addition, the advantage of both BRKGA-type heuristics compared with the ETH(FFBF-SS) procedure steadily diminishes up to around 2% for instances with $N = 240$, while the ILS scheme still provides around 3% of improvement. It can be observed that the performance depends on the number of jobs as expected due to the job-based representation. The advantage of the BRKGA-type approaches is larger when applied to instances with small order sizes (ν=3) and large FOUP capacity (β=2). The order-to-job assignment is harder for instances of this subset. At the same time, a smaller number of jobs is available which makes the search space of the BRKGA smaller.

The Appendix Tables A3 and A4 (supplementary material) contain the corresponding results for

Table 7. Results of the Wilcoxon signed-rank test for the lot processing mode.

	BRKGA-FR	ILS	BRKGA-JR
ETH(FFBF-LS)	>	>	>
BRKGA-FR		>	>
ILS			>

Table 8. Results of the Wilcoxon signed-rank test for the item processing mode.

	BRKGA-IJR	ILS	BRKGA-IOR	BRKGA-IOET
ETH(FFDAJS-LS)	>	>	>	>
BRKGA-IJR		>	>	>
ILS			>	>
BRKGA-IOR				>

problem (5). We observe that the application of BRKGA-type heuristics leads to significantly better solutions compared with the reference heuristic ETH(FFDAJS-LS). The advantage increases when the m values get larger while the L values have only a minor impact. The BRKGA-IOET outperforms the remaining algorithms under most experimental conditions. Note that this heuristic is the only one that is explicitly designed to respect Property 9. Therefore, it provides high-quality schedules especially for small-sized problem instances. The E/T values found are on average almost 50% smaller than the ones found with the ETH(FFDAJS-LS) procedure for instances with $N = 30$. Due to the DOE, there are instances with optimal schedules where all orders are early or on time. Note that for the factor combination $N = 30$, $m = 6$, $L = 3$, ν=3, and β=2 all orders are assigned to jobs that are completed at d in all obtained schedules. Only the BRKGA-IOET is able to determine these solutions. The advantage of the BRKGA-IOET decreases for medium- and large-sized instances. However, the obtained E/T values are still around 6% below the ones found by the simple reference heuristic. Again, the performance of the ILS scheme is between the performance of the best performing BRKGA-type algorithms and the ETH(FFDAJS-LS). The results of the Wilcoxon signed-rank test for problem (5) are reported in Table 8.

6. Conclusions and future research

We studied a parallel-machine MOJ scheduling problem with E/T measure. The orders have an unrestrictively late common due date. Only orders with identical family can be grouped into one job. Both the lot and the item processing mode were studied. The considered scheduling problems are important for next-generation wafer fabs where it is likely that MOJ problems will occur due to the increasing wafer size. They can be considered as model problems for just-in-time-type situations which are important in semiconductor manufacturing due to

high capital commitment costs in wafer fabs and the need for high on-time delivery performance. We proposed a MILP formulation for the two scheduling problems. We proved the NP-hardness for the scheduling problem with item processing mode whereas the NP-hardness in the lot processing mode is known from the literature. We then showed several structural properties for certain classes of optimal solutions. Because of the NP-hardness we designed list scheduling-based reference heuristics, an ILS scheme, and several BRKGA-based procedures based on different decomposition strategies. The heuristics exploit the properties proved before. We compared the different heuristics based on randomly generated problem instances. It turned out that some of the BRKGA-type heuristics perform well with respect to solution quality and required computing time.

There are several directions for future research. First of all, the grouping genetic algorithm (GGA) proposed by Sobeyko and Mönch (2015) for single-machine MOJ scheduling problems is an interesting alternative to the BRKGA-type approaches from this article. The GGA can form the jobs while the properties in this article can be used to assign the jobs to machines and sequence them there. Moreover, since the ILS is able to provide high-quality solutions for large-sized problem instances of the lot processing environment it is desirable to investigate the ILS scheme briefly sketched in this article in more detail. It seems possible to use more advanced neighbourhood structures and to apply the MOJAP.

As a third avenue for research, it is desirable to study extensions of the present problem setting to uniform or even unrelated machine environments. Moreover, this has to be combined with eligibility restrictions which are also common in most wafer fabs. Since wafer fabs are complex job shops, extensions towards flexible flow shops and job shops are also desirable. It seems interesting to allow a restrictive common due date for all the orders. To the best of our knowledge even in the p-batch situation this problem is not considered so far. We expect that most of the properties for optimal schedules are not valid anymore in this situation. Finally, replacing the common due date by a common due window similar to (Zhao et al., 2006) is also interesting since due windows can be often found in real-world settings.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Agrawal, G. K., & Heragu, S. S. (2006). A survey of automated material handling systems in 300-mm semiconductor fabs. *IEEE Transactions on Semiconductor Manufacturing*, *19*(1), 112–120. doi:10.1109/TSM.2005.863217

Azizoglu, M., & Webster, S. (1997). Scheduling job families about an unrestricted common due date on a single machine. *International Journal of Production Research*, *35*(5), 1321–1330. doi:10.1080/002075497195344

Bean, J. C. (1994). Genetic algorithm and random keys for sequencing and optimization. *ORSA Journal of Computing*, *6*(2), 154–160. doi:10.1287/ijoc.6.2.154

brkgaAPI (2018). Brkga source code. Retrieved from https://github.com/rfrancotoso/brkgaAPI/.

Conover, W. J. (1980). *Practical nonparametric statistics*. New York: John Wiley.

Emmons, H. (1987). Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics*, *34*(6), 803–810. doi:10.1002/1520-6750(198712)34:6<803::AID-NAV3220340605>3.0.CO;2-2

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.

Gonçalves, J. F., & Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, *17*(5), 487–525. doi:10.1007/s10732-010-9143-1

Gordon, V., Proth, J.-M., & Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, *139*(1), 1–25. doi:10.1016/S0377-2217(01)00181-3

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287–326. doi:10.1016/S0167-5060(08)70356-X

Hall, N. G. (1986). Single- and multiple processor models for minimizing completion time variance. *Naval Research Logistics Quarterly*, *33*(1), 49–54. doi:10.1002/nav.3800330105

Hall, N. G., & Posner, M. E. (1991). Earliness-tardiness scheduling problems I: weighted deviation of completion times about a common due date. *Operations Research*, *39*(5), 836–846. doi:10.1287/opre.39.5.836

Jampani, J., & Mason, S. J. (2008). Column generation heuristics for multiple machine, multiple orders per job scheduling problems. *Annals of Operations Research*, *159*(1), 261–273. doi:10.1007/s10479-007-0281-2

Jampani, J., Pohl, E. A., Mason, S. J., & Mönch, L. (2010). Integrated heuristics for scheduling multiple order jobs in a complex job shop. *International Journal of Metaheuristics*, *1*(2), 156–180. doi:10.1504/IJMHEUR.2010.034204

Jia, J., & Mason, S. J. (2009). Semiconductor manufacturing scheduling of jobs containing multiple orders on identical parallel machines. *International Journal of Production Research*, *47*(10), 2565– 2585. doi:10.1080/00207540701725042

Jozefowska, J. (2007). *Just-in-time scheduling: models and algorithms for computer and manufacturing systems*. New York: Springer.

Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, *28*(4), 643–651. doi:10.1002/nav.3800280411

Kim, J.-G., Kim, J.-S., & Lee, D.-H. (2012). Fast and meta-heuristics for common due-date assignment and scheduling on parallel machines. *International Journal of Production Research*, *50*(20), 6040–6057. doi:10.1080/00207543.2011.644591

Laub, J. D., Fowler, J. W., & Keha, A. B. (2007). Minimzing makespan with multiple-orders-per-job in a two-machine flowshop. *European Journal of Operational Research*, *128*(1), 63–79. doi:10.1016/j.ejor.2006.07.023

Lauff, V., & Werner, F. (2004). Scheduling with common due date, earliness and tardiness penalties for multimachine problems. *Mathematical and Computer Modelling*, *40*(5–6), 637–655. doi:10.1016/j.mcm.2003.05.019

Li, X., Chen, H., Xu, R., & Li, X. (2015). Earliness–tardiness minimization on scheduling a batch processing machine with non-identical job sizes. *Computers & Industrial Engineering*, *87*, 590–599. doi:10.1016/j.cie.2015.06.008

Mason, S. J., Qu, P., Kutanoglu, E., & Fowler, J. W. (2004). *The single machine multiple orders per job scheduling problem* (Technical Report No. ASUIE-ORPS-2004-04). Tempe: Arizona State University.

Mathirajan, M., & Sivakumar, A. I. (2006). Literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology*, *29*(9-10), 990–1001. doi:10.1007/s00170-005-2585-1

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Berlin: Springer.

Mönch, L. (2002). A genetic algorithm heuristic applied to stepper scheduling. In G. T. Mackulak, J. W. Fowler, and A. Schömig (Eds.), *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2002)* (pp. 276–281), Tempe, USA.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., & Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, *14*(6), 583–595. doi:10.1007/s10951-010-0222-9

Mönch, L., Fowler, J. W., & Mason, S. J. (2013). *Production planning and control for wafer fabrication facilities: modeling, analysis, and systems*. New York: Springer.

Mönch, L., & Roob, S. (2018). A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing*, *68*, 835–846. doi:10.1016/j.asoc.2017.10.028

Mönch, L., & Unbehaun, R. (2007). Decomposition heuristics for minimizing earliness-tardiness on parallel burn-in ovens with a common due date. *Computers & Operations Research*, *34*(11), 3380–3396. doi:10.1016/j.cor.2006.02.003

Mönch, L., Unbehaun, R., & Choung, Y. I. (2006). Minimizing earliness and tardiness on a single burn-in oven with a common due date and a maximum available tardiness constraint. *Or Spectrum*, *28*(2), 177–198. doi:10.1007/s00291-005-0013-4

Ogun, B., & Alabas-Uslu, C. (2018). Mathematical models for a batch scheduling problem to minimize earliness and tardiness. *Journal of Industrial Engineering and Management*, *11*(3), 390–405. doi:10.3926/jiem.2541

Parsa, N. R., Karimi, B., & Moattar Husseini, S. M. (2017). Exact and heuristic algorithms for the just-in-time scheduling problem in a batch processing system. *Computers & Operations Research*, *80*, 173–183. doi:10.1016/j.cor.2016.12.001

Pei, J., Cheng, B., Liu, X., Pardalos, P. M., & Kong, M. (2019). Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time. *Annals of Operations Research*, *272*(1-2), 217–241. doi:10.1007/s10479-017-2481-8

Pei, J., Liu, X., Pardalos, P. M., Fan, W., & Yang, S. (2017a). Scheduling deteriorating jobs on a single serial-batching machine with multiple job types and sequence-dependent setup times. *Annals of Operations Research*, *249*(1-2), 175–195. doi:10.1007/s10479-015-1824-6

Pei, J., Liu, X., Pardalos, P. M., Migdalas, A., & Yang, S. (2017b). Serial-batching scheduling with time-dependent setup time and effects of deterioration and learning on a single-machine. *Journal of Global Optimization*, *67*(1-2), 251–262. doi:10.1007/s10898-015-0320-5

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: a review. *European Journal of Operational Research*, *120*(2), 228–249. doi:10.1016/S0377-2217(99)00153-8

Rocholl, J., & Mönch, L. (2017). Hybrid heuristics for multiple orders per job scheduling problem with parallel machines and a common due date. In A. Gunawan, G. Kendall, L. S. Lee, B. McCollum, & H.-V. Seow (Eds.) *Proceedings MISTA 2017*. (pp. 358–361). Kuala Lumpur, Malaysia.

Rocholl, J., & Mönch, L. (2018). Hybrid algorithms for the earliness-tardiness single-machine multiple orders per job scheduling problem with a common due date. *Rairo - Operations Research*, *52*(4-5), 1329– 1350. doi:10.1051/ro/2018029

Sarin, S. C., Wang, L., & Cheng, M. (2012). Minimising makespan for a two-machine, flow shop, single-wafer-processing, multiple-jobs-per-carrier scheduling problem. *International Journal of Planning and Scheduling*, *1*(3), 171–208. doi:10.1504/IJPS.2012.050126

Semi. (2018). *450mm*. Retrieved from http://www1.semi.org/en/450mm.

Sobeyko, O., & Mönch, L. (2015). Grouping genetic algorithms for solving single machine multiple orders per job scheduling problems. *Annals of Operations Research*, *235*(1), 709–739. doi:10.1007/s10479-015-1976-4

Suriyaarachchi, R. H., & Wirth, A. (2004). Earliness/tardiness scheduling with a common due date and family setups. *Computers & Industrial Engineering*, *47*, 275–288. doi:10.1016/j.cie.2004.07.006

Toso, R. F., & Resende, M. G. C. (2011). A C++ application programming interface for biased random-key genetic algorithms. Retrieved from http://mauricio.resende.info/doc/brkgaAPI.pdf.

Voß, S., & Woodruff, D. L. (2006). *Introduction to computational optimization models for production planning in a supply chain* (2nd ed.). New York: Springer.

Yugma, C., Dauzère-Pérès, S., Artigues, C., Derreumaux, A., & Sibille, O. (2012). A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. *International Journal of Production Research*, *50*(8), 2118–2132. doi:10.1080/00207543.2011.575090

Zhao, H., Hu, F., & Li, G. (2006). Batch scheduling with a common due window on a single machine. In *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery 2006*. LNCS 4223, 641–645.