Modelling and symmetry breaking in scheduling problems on batch processing machines

Renan Spencer Trindade, Olinto César Bassi de Araújo, Marcia Helena Costa Fampa, Felipe Martins Müller

Symmetry breaking constraints and An arc-flow formulation for scheduling problems on batch processing machines

Renan Spencer Trindade*[1], Olinto C. B. de Araújo[1], Marcia Fampa[2], and Felipe Martins Müller[3]

[1] *Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brazil*
[2] *Colégio Técnico Industrial de Santa Maria, Universidade Federal de Santa Maria, Brazil*
[3] *Departamento de Eletrônica e Computação, Universidade Federal de Santa Maria, Santa Maria, RS, Brazil*

Abstract

Problems of scheduling batch-processing machines to minimize the makespan are widely exploited in the literature, mainly motivated by real-world applications, such as burn-in tests in the semiconductor industry. These problems consist of grouping jobs in batches and scheduling them on machines. We consider problems where jobs have non-identical sizes and processing times, and the total size of each batch cannot exceed the machine capacity. The processing time of a batch is defined as the longest processing time among all jobs assigned to it. Jobs can also have non-identical release times, and in this case, a batch can only be processed when all jobs assigned to it are available. This paper discusses four different versions of batch scheduling problems, considering a single processing machine or parallel processing machines, and considering jobs with or without release times. New mixed integer linear programming formulations are proposed as enhancements of formulations proposed in the literature, and symmetry breaking constraints are investigated to reduce the size of the feasible sets. Computational results show that the proposed formulations have a better performance than other models in the literature, being able to solve to optimality instances only considered before to be solved by heuristic procedures.

Keywords: Batch processing machine; BPM; Symmetry; Makespan; Mixed integer linear program

1 Introduction

Scheduling problems in batch processing machines have been extensively explored in the literature, motivated by a large number of applications in industries and also by the challenging solution of real world problems. The main goal in these problems is to group jobs in batches and process them simultaneously in a machine, to facilitate the tasks and to reduce the time spent in handling the material. Although there are many variations of the problem involving batch processing machines,

*Corresponding author. Email: trindade@cos.ufrj.br

This is an Accepted Manuscript version of the following article, accepted for publication in International Journal of Production Research. Trindade, R. S., de Araújo, O. C. B., Fampa, M. H. C., & Müller, F. M. (2018). Modelling and symmetry breaking in scheduling problems on batch processing machines. International Journal of Production Research, 56(22), 7031–7048. DOI: 10.1080/00207543.2018.1424371.

the versions addressed in this work are more suitable to model the scheduling problems that arise in reliability tests in the semiconductor industry, in operations called burn-in, presented in [22].

The burn-in operation is used to test electronic circuits and consists of designating them to industrial ovens, submitting them to thermal stress for a long period. The test of each circuit is considered here as a job and requires a minimum time inside the oven, which is referred to as the processing machine. The jobs cannot be processed directly on the machine, they need to be placed on a tray, respecting the capacity of the machine. Each group of jobs assigned to a tray is considered a batch. The minimum time of the circuit inside the oven is set a priori, based on the supplier requirements. It is possible to keep the circuit in the oven longer than necessary, with no prejudice, but it cannot be removed before its required processing time is fulfilled. Therefore, the processing time of a batch is determined by the longest processing time among all jobs assigned to it. The burn-in tests are a bottleneck in final testing operations, and the efficient scheduling of these operations aims to maximize productivity and reduce flow time in the stock, which is a major concern for management.

For the problems addressed in this paper, we are given a set of jobs. Each job j has a size s_j, a minimum processing time p_j, and must be assigned to a batch. The sum of the sizes of the jobs assigned to a batch cannot exceed the capacity limit B of the processing machine. The batches must be scheduled on a machine, which can process only one batch at a time. The goal is to design and schedule the batches on the machines so that the completion time of the last job processed, called makespan and denoted by $C_{\max}$, is minimized. Finally, it is also possible that each job j has a different release time r_j, which is when the job becomes available to be processed. In this case, a batch can only be processed when all the jobs assigned to it are available.

We address four different versions of this scheduling problem. On the two first, denoted by $1|s_j, B|C_{\max}$ and $P_m|s_j, B|C_{\max}$, we do not consider different release times for the jobs. On problem $1|s_j, B|C_{\max}$, only a single processing machine is used, and on $P_m|s_j, B|C_{\max}$, m parallel identical machines are used. The two other problems, denoted by $1|r_j, s_j, B|C_{\max}$ and $P_m|r_j, s_j, B|C_{\max}$, are defined equivalently to the two first ones, but with non-identical release times considered for the jobs.

In this paper, we first present mixed integer linear programming (MILP) formulations for the four problems addressed, which were proposed in Melouk, Damodaran and Chang [19], [2], [24] and [23]. We point out that all the four formulations present a large number of symmetric solutions in their feasible sets. Two types of symmetries are considered in our analysis. On the first one, two solutions are said to be symmetric if the designs of the batches are equal on both solutions and the batches assigned to a machine are processed in the same order. On the second, two solutions are said to be symmetric if the designs of the batches are still equal in both solutions, but the batches assigned to at least one machine are processed in a different order. Nevertheless, the modification in the order in which the batches are processed does not affect the makespan, and therefore also generates equivalent solutions. The existence of symmetric solutions in the feasible set of the problems leads to a very inefficient application of branch-and-bound (B&B) algorithms. Treatment of symmetry in integer programs is an intense area of research, where different strategies are suggested to mitigate the effect of symmetric solutions during the B&B execution (see, for example, [18]). The impact of symmetry breaking constraints on a particular software engineering application is also investigated in [16]. In this work, several sets of symmetric solutions for the problem addressed are pointed out, as well as symmetry breaking constraints to deal with them. These constraints motivated the study of symmetry breaking for the scheduling problems addressed in this paper. Furthermore, we take into account specific properties of the problems and their optimal solutions to propose new stronger formulations for them and avoid undesirable symmetric solutions in their feasible sets. We show the correctness of our models and explain how our different indexing choices for each problem allow a more efficient modeling. Applying our models we are able to prove optimality of instances with sizes considered for the first time in the literature to be solved by exact methods.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the main results that we found in the literature, on the problems addressed. In Section 3, we present the MILP formulations from the literature. In Section 4, we analyze symmetric solutions to these formulations and propose symmetry breaking constraints and new formulations for the problems.

In Section 5, we present computational results comparing the different formulations. In Section 6, we present some concluding remarks.

2 Literature Review

Problem $1|s_j, B|C_{\max}$ was addressed for the first time in [22], where this NP-hard complexity is proved and a heuristic approach to solve it is proposed. Heuristics are also proposed for this problem in [12], where instances with up to 100 jobs are considered. Two approximation algorithms are presented in [25] with approximation ratios of 3/2 and 7/4 of the optimal solution, in the worst case. In [19], the simulated annealing meta-heuristic was applied to $1|s_j, B|C_{\max}$ and an MILP formulation was presented for the problem. This work also proposes configurations for test instances that were widely used in later works. Computational results are shown for instances with up to 100 jobs, comparing the heuristic solutions to the solutions obtained with the MILP formulation. Other meta-heuristics are also applied to problem $1|s_j, B|C_{\max}$ in the literature, namely, genetic algorithm ([9] and [14]), tabu search ([20]), and GRASP ([8]). These four papers consider instances with up to 100 jobs as well. In addition, the bee colony meta-heuristic is also applied to the problem in [1], where results for instances with up to 200 jobs are shown. In [3], a heuristic based on a special case of the clustering problem is proposed, and test instances with up to 500 jobs are considered. In [17], two heuristics are proposed based on a decomposition of the original problem, where relaxations of the problem are solved. Instances with up to 100 jobs are considered in this work. An exact approach is used in [21], where a formulation for problem $1|s_j, B|C_{\max}$ is presented, using a partition problem in the context of Dantzig-Wolfe decomposition. In this work, the branch-and-price method is applied to solve the problem to optimality. Instances with up to 500 jobs are considered in the computational experiments.

Concerning problem $P_m|s_j, B|C_{\max}$, the papers that address it, are mostly extensions of the works published for problem $1|s_j, B|C_{\max}$. In [2], the simulated annealing meta-heuristic is applied, and an MILP formulation is presented for the problem. This work also proves the NP-hard complexity of the problem, and show results for instances with up to 50 jobs. In [15], a hybrid genetic algorithm is used to compute solutions for instances with up to 100 jobs, considering 2 and 4 parallel machines. In [10] a new application of the genetic algorithm is proposed, which solves instances with up to 100 tasks, also on 2 and 4 parallel machines. In [4] an approximation algorithm is presented for the problem, with the approximation factor of 2. Finally, two other works that apply meta-heuristics ([5] and [13]), use the ant colony method and a meta-heuristic based on a max-min ant system for this problem. In [5], results for instances with up to 500 jobs on 4 and 8 parallel machines are shown, whereas, in [13], instances are solved with up to 100 jobs, on 2, 3, and 4 parallel machines.

There are only a few papers investigating problem $1|r_j, s_j, B|C_{\max}$ in the literature. To our knowledge, only three papers have been published. Solution approaches based on meta-heuristics are presented in [6] and [24], which apply hybrid genetic algorithms and the ant colony meta-heuristic, respectively. In both works, test instances with up to 100 jobs are considered. In [26], three heuristics are proposed for this problem and computational results for instances with up to 300 jobs are shown. In [24], we can also find an MILP formulation for the problem.

Problem $P_m|r_j, s_j, B|C_{\max}$ was firstly addressed in [7]. In this paper, the authors propose an MILP formulation to solve the problem to optimality and three heuristics to handle instances with 7 and 15 jobs. In [23], the problem is proved to be NP-hard, and is addressed with the use of an MILP formulation and also with five heuristics and with the application of two meta-heuristics, simulated annealing, and GRASP. It is also proposed a column generation method, which generates lower bounds to the optimal solution of the problem by solving its continuous relaxation. Numerical experiments with instances with up to 50 jobs, and with 3 and 5 parallel machines, are presented. The results of this work are also published in [11].

We can conclude from our literature review on the problems addressed in this work, that most of the effort made by researchers to solve them, concentrated in heuristic procedures. The MILP formulations presented for them were mostly used as a baseline to give a formal definition of the problems, and provide some evaluation for the heuristic solutions on small instances. As mentioned

before, B&B algorithms become very inefficient when applied to these formulations due to the presence of symmetric solutions in their feasible sets.

3 Description of the problems addressed

3.1 Problem $1|s_j, B|C_{\max}$

Problem $1|s_j, B|C_{\max}$ is the simplest job scheduling problem addressed in this paper. It can be formally defined as follows. Given a set $J := \{1, \ldots, n_J\}$ of jobs, each job $j \in J$ has a processing time p_j and a size s_j. Each of them must be assigned to a batch $k \in K := \{1, \ldots, n_K\}$, not exceeding a given capacity limit B of the processing machine, i.e., the sum of the sizes of the jobs assigned to a single batch cannot exceed B. We assume that $s_j \leq B$, for all $j \in J$. The batches must be all processed in a single machine, one at a time, and all the jobs assigned to a single batch are processed simultaneously. The processing time P_k of each batch $k \in K$ is defined as longest processing time among all jobs assigned to it, i.e., $P_k := \max\{p_j : j \text{ is assigned to } k\}$. Jobs cannot be split between batches. It's also not possible to add or remove jobs from the machine while the batches are being processed. The goal is to design and schedule the batches so that the makespan ($C_{\max}$) is minimized, where the design of a batch is defined as the set of jobs assigned to it, to schedule the batches means to define the ordering in which they are processed in the machine, and the makespan is defined as the time required to finish processing the last batch. The number of batches used on the solution is not fixed and should be optimized. It will depend on the number of jobs, their sizes, and the machine capacity. In the worst case, the number of batches will be equal to the number of jobs. Therefore, to assure the correctness of the formulations presented in this section, it is assumed that $n_K = n_J$.

Let's consider then the following decision variables for all $j \in J$, $k \in K$:

$$x_{jk} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$y_k = \begin{cases} 1, & \text{if batch } k \text{ is used;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$P_k : \text{processing time of batch } k. \tag{3}$$

In [19] the following MILP formulation is proposed for problem $1|s_j, B|C_{\max}$. Other very similar formulations and sometimes this exactly same one, are used in other papers as a comparative basis in computational experiments.

$$(\text{MILP}_1) \quad \min \sum_{k \in K} P_k, \tag{4}$$

$$\sum_{k \in K} x_{jk} = 1, \qquad \forall j \in J, \tag{5}$$

$$\sum_{j \in J} s_j x_{jk} \leq B y_k, \qquad \forall k \in K, \tag{6}$$

$$P_k \geq p_j x_{jk}, \qquad \forall j \in J, \forall k \in K, \tag{7}$$

$$x_{jk} \leq y_k, \qquad \forall j \in J, \forall k \in K, \tag{8}$$

$$P_k \geq 0, \qquad \forall k \in K, \tag{9}$$

$$y_k \in \{0, 1\}, \qquad \forall k \in K, \tag{10}$$

$$x_{jk} \in \{0, 1\}, \qquad \forall j \in J, \forall k \in K. \tag{11}$$

The objective function (4) minimizes the makespan, given by the sum of the processing times of all batches. Constraints (5) determine that each job is assigned to a single batch. Constraints (6) determine that each batch if used does not exceed the capacity of the machine. Constraints (7) determine the processing times of the batches. Note that constraints (8) are redundant together with (6), but are added to strengthen the linear relaxation of the formulation.

3.2 Problem $P_m|s_j, B|C_{\max}$

Problem $P_m|s_j, B|C_{\max}$ is very similar to problem $1|s_j, B|C_{\max}$. The only difference is the possibility of using more than one machine to process the batches. When defining problem $P_m|s_j, B|C_{\max}$, besides considering all aspect presented to $1|s_j, B|C_{\max}$, we also assume that the batches should be assigned to a specific machine $m \in M := \{1, \ldots, n_M\}$. All machines are identical and the objective is again to minimize the makespan ($C_{\max}$), which is now defined as the time required to finish processing the last batch in all machines.

Consider the following decision variables, for all $j \in J$, $k \in K$, and $m \in M$:

$$x_{jkm} = \begin{cases} 1, & \text{if job } j \text{ is assigned to batch } k \text{ processed in machine } m; \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

$$P_{km} : \text{time to process batch } k \text{ in machine } m. \tag{13}$$

In [2] the following MILP formulation is proposed for $P_m|s_j, B|C_{\max}$:

$$(\text{MILP}_2) \quad \min \ C_{\max}, \tag{14}$$

$$\sum_{k\in K} \sum_{m\in M} x_{jkm} = 1, \qquad \forall j \in J, \tag{15}$$

$$\sum_{j\in J} \sum_{m\in M} s_j x_{jkm} \leq B, \qquad \forall k \in K, \tag{16}$$

$$P_{km} \geq p_j x_{jkm}, \qquad \forall j \in J, \forall k \in K, \forall m \in M, \tag{17}$$

$$C_{\max} \geq \sum_{k\in K} P_{km}, \qquad \forall m \in M, \tag{18}$$

$$P_{km} \geq 0, \qquad \forall k \in K, \forall m \in M, \tag{19}$$

$$C_{\max} \geq 0, \tag{20}$$

$$x_{jkm} \in \{0, 1\}, \qquad \forall j \in J, \forall k \in K, \forall m \in M. \tag{21}$$

The objective function (14) minimizes the makespan. Constraints (15) and (16) ensure that each job is assigned to a single batch and a single machine, respecting the capacity of the machine. Constraints (17) determine the processing time of batch k in machine m. Constraints (18) determine the makespan, which is given by the longest sum of the processing times of all batches, among all machines.

Note that formulation (MILP_2) takes into account that $n_K = n_J$, and therefore, all batches assigned to all machines on a given solution can be indexed by distinct indexes.

3.3 Problem $1|r_j, s_j, B|C_{\max}$

Problem $1|r_j, s_j, B|C_{\max}$ is also very similar to problem $1|s_j, B|C_{\max}$. The only difference now is that jobs have non-identical release times. All other aspects are similar to the ones presented for problem $1|s_j, B|C_{\max}$. In $1|r_j, s_j, B|C_{\max}$, we admit that each job $j \in J$ has a release time r_j and can only be processed after released. Accordingly, each batch can only be processed when all jobs assigned to it have been released. The release time of batch k is then defined as $R_k := \max\{r_j : j \text{ is assigned to } k\}$.

Let's consider now the decision variables (1), (3), and also the variables:

$$S_k : \text{time when batch } k \text{ starts to be processed,} \tag{22}$$

for all $k \in K$.

In [24] the following MILP formulation is proposed for $1|r_j, s_j, B|C_{\max}$. We emphasize the modifications made to formulation (MILP_1), which was previously presented for $1|s_j, B|C_{\max}$.

$$\text{(MILP}_3\text{)} \quad \min\ S_{n_K} + P_{n_K}, \tag{23}$$

$$\sum_{j \in J} s_j x_{jk} \leq B, \qquad \forall k \in K, \tag{24}$$

$$S_k \geq r_j x_{jk}, \qquad \forall j \in J, \forall k \in K, \tag{25}$$

$$S_k \geq S_{k-1} + P_{k-1}, \qquad \forall k \in K : k > 1, \tag{26}$$

$$S_k \geq 0, \qquad \forall k \in K, \tag{27}$$

$$(5),\ (7),\ (9),\ (11). \tag{28}$$

The objective function (23) minimizes the makespan, given by the starting time of the last batch processed added to its processing time. Constraints (24) ensure that each batch respects the capacity of the machine. Constraints (25)-(27) determine the time when each batch starts to be processed. The other constraints were taken from (MILP$_1$). Note that the binary variables y_k, defined in (2), and previously used to strengthen the linear relaxation of problem $1|s_j, B|C_{\max}$, are not used in this formulation.

3.4 Problem $P_m|r_j, s_j, B|C_{\max}$

Problem $P_m|r_j, s_j, B|C_{\max}$ is very similar to $P_m|s_j, B|C_{\max}$. Once more, the only difference is that jobs now have non-identical release times. All other aspects are similar to the ones presented for problem $P_m|s_j, B|C_{\max}$. As in $1|r_j, s_j, B|C_{\max}$, the jobs can only be processed after released in problem $P_m|r_j, s_j, B|C_{\max}$. The objective is again to minimize the makespan ($C_{\max}$), defined as the time required to finish processing the last batch in all machines.

Let's consider now the decision variables (12), and also the variables:

$$S_{km} : \text{time when batch } k \text{ starts to be processed in machine } m, \tag{29}$$

for all $k \in K$, and $m \in M$.

In [23] the following MILP formulation is proposed for $P_m|r_j, s_j, B|C_{\max}$. We emphasize the modifications made to formulation (MILP$_2$) previously presented for $P_m|s_j, B|C_{\max}$.

$$\text{(MILP}_4\text{)} \quad \min\ C_{\max}, \tag{30}$$

$$\sum_{j \in J} s_j x_{jkm} \leq B, \qquad \forall k \in K, \forall m \in M \tag{31}$$

$$S_{km} \geq r_j x_{jkm}, \qquad \forall j \in J, \forall k \in K, \forall m \in M, \tag{32}$$

$$S_{km} \geq S_{(k-1)m} + p_j x_{j(k-1)m}, \qquad \forall j \in J, \forall k \in K : k > 1, \forall m \in M, \tag{33}$$

$$S_{km} \geq 0, \qquad \forall k \in K, \forall m \in M, \tag{34}$$

$$C_{\max} \geq S_{n_K m} + p_j x_{j n_K m}, \qquad \forall m \in M, \tag{35}$$

$$(15),\ (20),\ (21). \tag{36}$$

The objective function (30) minimizes the makespan. Constraints (31) ensure that each batch respects the capacity of the machine. Constraints (32)-(34) determine the time when each batch k starts to be processed in each machine m. Constraints (35) determine the makespan. The other constraints are taken from (MILP$_2$).

4 Symmetry breaking

The four formulations presented in the previous section allow a large number of symmetric solutions in the feasible sets of the problems. We distinguish two types of symmetry that may occur in all formulations. On the first, solutions where the batches are all equally designed, are identically assigned to machines, and are also processed in the machines in the same order, may be represented as different solutions and coexist in the feasible set. This happens whenever the number of batches actually used or processed in a given machine is smaller than the number of jobs n_J. Clearly, this

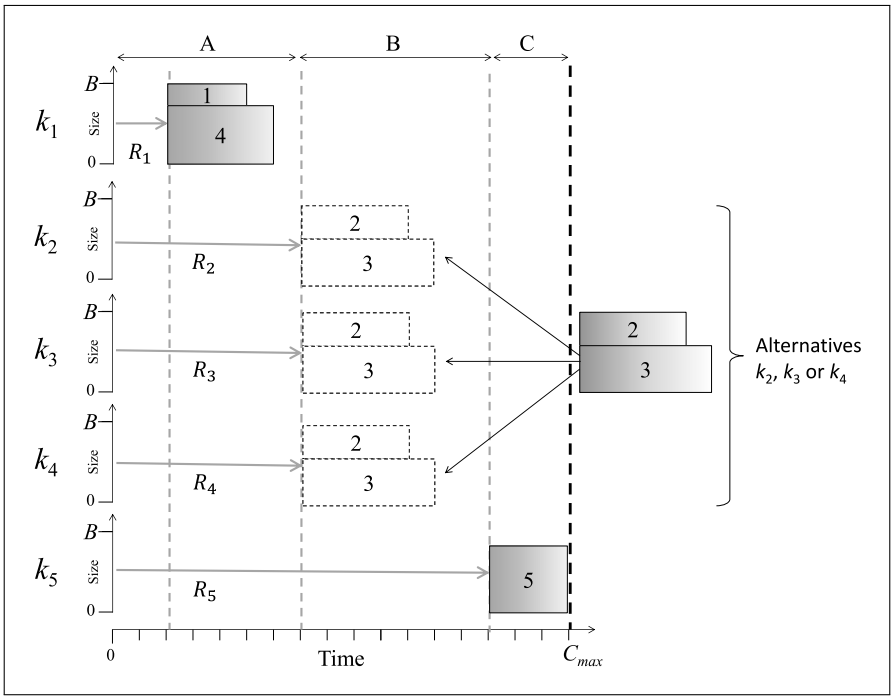


Figure 1: Symmetric solutions in which batches are represented by different indexes.

allows the solutions to be represented by different indexing of the batches, as illustrated in Figure 1, where three symmetric solutions for a problem with five jobs, equally grouped in three batches, are depicted. Note that the only difference between the solutions is the indexing of the batch represented as B. Its index could be k_2, k_3, or k_4. The four formulations presented in the previous section allow this type of symmetry.

The second symmetry analyzed occurs when changing the processing order of the batches assigned to a machine, does no affect the makespan. Surely, this always happens on problems where all jobs have identical release times. In this case, the processing time of a machine is given by the sum of the processing times of the batches assigned to it, the machine is never idle during their processing, and any permutation in the order in which those batches are processed lead to equivalent solutions, with the same makespan. This situation is exemplified in Figure 2, which depicts two equivalent solutions, (a) and (b), for an instance of the problem with 10 jobs. On both solutions all batches are equally designed, having the same jobs assigned to them. The makespan is consequently also the same for both solutions. The only difference between them is the sequence in which the batches are processed in the machine. Note that any other permutation would lead to another equivalent solution.

It is important to note, however, that when non-identical release times are considered for the jobs, it is not assured anymore that permutations in the processing order of the batches will still lead to equivalent solutions, as it may affect the idle time of the machine, and consequently change the makespan. Therefore, when eliminating different ordering possibilities from the feasible set, we should make sure the optimal ordering is not cut off. This point will be better explored below.

Our goal in this section is to present new formulations for the problems addressed, where these two types of symmetry are eliminated from the feasible set of all problems. We deal with the symmetries in two different ways, which take into account whether or not the jobs have non-identical release times. In either case, we propose an indexing for the given set of jobs that, together with some analysis of properties of the optimal solutions, allows a significant improvement on the formulations presented in Section 3.

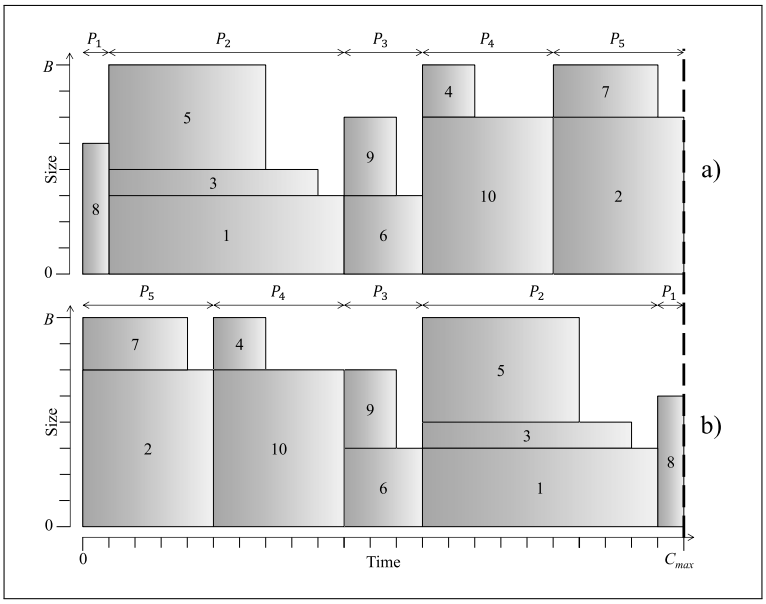


Figure 2: Symmetric solutions for problem $1|s_j, B|C_{\max}$.

4.1 Case where jobs have identical release-times

Although problem $1|s_j, B|C_{\max}$ admit only one processing machine, it can already be considered highly symmetric. Besides symmetries of the first type described above, all permutations in the processing order of the batches scheduled in the single machine, also lead to equivalent solutions. In model (MILP$_1$) all these solutions are considered distinct from each other, generating a large feasible set with many equivalent solutions.

We deal with the symmetry of problem $1|s_j, B|C_{\max}$ with a threefold procedure. Firstly, we set the indexes of the jobs, ordering them by their processing time. More specifically, we consider:

$$p_1 \leq p_2 \leq \ldots \leq p_{n_J}. \tag{37}$$

Secondly, we set $n_K := n_J$ and determine that batch k can only be used if job k is assigned to it, for all $k \in K$. Thirdly, we determine that job j can only be assigned to batch k if $j \leq k$.

Note that with the assumptions made, we may only define variables x_{jk} in (1), for $j \leq k$, reducing the number of binary variables form n_J^2 to $n_J(n_J+1)/2$. More importantly, the assumptions lead to solutions where the processing time of batch k, if used, is equal to p_k, as job k is certainly assigned to it, and is also the job with longest processing time assigned to the batch. Consequently, there is no need of defining the variables P_k (3), for all $k \in K$, in order to represent the processing times of the batches, neither of imposing constraints (7) to determine them.

Besides reducing the number of variables, when compared to (MILP$_1$), the strategy described leaves only one possible processing ordering for the batches in a given solution, where the batches are ordered by non-decreasing processing time. Furthermore, there is only one possible way of idexing the batches on a given solution, where the index of the batch is equal to the largest index among the jobs assigned to it. Several equivalent solutions are therefore, eliminated from the feasible set of (MILP$_1$).

Considering the above, we propose next a new formulation for $1|s_j, B|C_{\max}$:

$$(\text{MILP}_1^+) \quad \min \sum_{k\in K} p_k x_{kk}, \tag{38}$$

$$\sum_{k\in K:k\geq j} x_{jk} = 1, \qquad \forall j \in J, \tag{39}$$

$$\sum_{j\in J:j\leq k} s_j x_{jk} \leq B x_{kk}, \qquad \forall k \in K, \tag{40}$$

$$x_{jk} \leq x_{kk}, \qquad \forall j \in J, \forall k \in K : j \leq k, \tag{41}$$

$$x_{jk} \in \{0,1\}, \qquad \forall j \in J, \forall k \in K : j \leq k. \tag{42}$$

The objective function (38) minimizes the makespan, given by the sum of the processing times of the batches used. Constraints (39) determine that each job j is assigned to a single batch k, such that $k \geq j$. Constraints (40) determine that the batches do not exceed the capacity of the machine. They also ensure that each batch k is used if and only if job k is assigned to it. Constraints (41) are redundant together with (40), but are included to strengthen the linear relaxation of the model.

It is straightforward to verify that the minimum makespan of (MILP_1) and (MILP_1^+) are the same. The following proposition formalizes this result.

Proposition 1. *The optimal makespan of problems ($MILP_1$) and ($MILP_1^+$) are the same.*

Proof. Let us consider w.l.o.g. that the indexes of jobs in J satisfy (37). Clearly, any feasible solution of (MILP_1^+) is also a feasible solution to (MILP_1) with the same objective function value. Therefore, it suffices to show that given any feasible solution to (MILP_1), with objective function value $\bar{C}_{\max}$, there is a feasible solution to (MILP_1^+) also with objective function value $\bar{C}_{\max}$. For that, let us first reset, if necessary, the indexes of the batches on the given solution of (MILP_1), determining them as the largest index among the ones of all jobs assigned to it. The solution is now a feasible to solution to (MILP_1^+), with the batches designed as in the given feasible solution to (MILP_1), but possibly processed in a different order. As this reordering of the batches does not affect the makespan in this problem, both solutions have the same objective function value. □

Considering now problem $P_m|s_j, B|C_{\max}$, we note that the same symmetry mentioned above for problem $1|s_j, B|C_{\max}$ is also present in this problem, and we can use a similar symmetry breaking procedure to the one described above. For modeling problem $P_m|s_j, B|C_{\max}$, we initially note that variables x_{jkm} in (12) determine the design of the batches and also assign them to a specific machine. We propose the replacement of these variables with the binary variables x_{jk}, which determine only the design of the batches, as defined in (1), and the binary variables y_{km}, which determine whether or not batch k is processed in machine m, for all j and m. This replacement significantly reduces the number of binary variables. In (MILP_4), the number of variables x_{jkm} is equal to $(n_J^2)(n_M)$. In our proposed model, we will have $n_J(n_J+1)/2$ variables x_{jk}, as in (MILP_1^+), plus $n_J n_M$ variables y_{km}. Furthermore, using the same procedure described for $1|s_j, B|C_{\max}$ to eliminate equivalent solutions from the feasible set of the $P_m|s_j, B|C_{\max}$, we next propose a new formulation for this problem.

$$(\text{MILP}_2^+) \quad \min\ C_{\max}, \tag{43}$$

$$(39),\ (40),\ (41),\ (42), \tag{44}$$

$$x_{kk} \leq \sum_{m\in M} y_{km}, \qquad \forall k \in K, \tag{45}$$

$$C_{\max} \geq \sum_{k\in K} p_k y_{km}. \qquad \forall m \in M, \tag{46}$$

$$C_{\max} \geq 0. \tag{47}$$

The objective function (43) minimizes the makespan given by the latest time to finish processing all batches in all machines. Constraints (45) ensure that each used batch is assigned to a machine. Constraints (46) and (47) determine the makespan. The other constraints are taken from (MILP_1^+).

4.2 Case where jobs have non-identical release-times

Unlike the previous case where jobs have identical release times, when the jobs have different release times, the order in which the batches are processed in a machine may modify the resulting makespan. The reason for this is the possible modification on the amount of time in which the machine stays idle, when permutations on the processing order are made.

Figures 3 depict two different solutions for a problem, where a permutation on the processing order of the batches leads to a different makespan. In the example, batches k_1 and k_2 shown in Figure 3(a) switch places in the processing order. The modification, shown in Figure 3(b), increases the idle time of the processing machine leading to an increase on the makespan.

Nevertheless, although a modification in the processing order of the batches leads in general, to nonequivalent solutions, symmetric solutions may still occur. Two cases are illustrated in Figure 4. In Case 1, two consecutive batches k and k' have the same release time. Thus, switching their indexes clearly does not affect the value of the makespan. In Case 2, batch k is released before k', but the machine is not idle at this time, due to the processing of the previously scheduled batch k''. In this configuration, switching k and k' does not affect the value of the makespan as well.

To deal with the symmetry of problem $1|r_j, s_j, B|C_{\max}$, we use a similar approach to the one proposed for problems where jobs have identical release times. However, now the jobs must be ordered by non-decreasing release times, and not by non-decreasing processing times as before. More specifically, we consider that the jobs are indexed satisfying:

$$r_1 \leq r_2 \leq \ldots \leq r_{n_J}. \tag{48}$$

As done for problems $1|s_j, B|C_{\max}$ and $P_m|s_j, B|C_{\max}$, the symmetry of problem $1|r_j, s_j, B|C_{\max}$ can be addressed by considering the variables x_{jk} in (1), only for $j \leq k$, together with the constraints (39)-(41), that assure that each job j is assigned to a single batch k, such that $j \leq k$. Once more, if batch k is used, we enforce job k to be the job with the highest index assigned to it. In this case, however, job k determines the release time of batch k, given by r_k, and not the processing time. Thus, the solutions presented by the formulation proposed, also present a specific ordering for the batches scheduled in the processing machine. The batches that are used, are processed in a non-decreasing order of their release times. Clearly, for a given set of designed batches, no other processing order for them would lead to a smaller makespan. Finally, for a given solution, the procedure allows only one possible way of indexing the batches, where the index of each batch is again given by the largest index among the jobs assigned to it.

We propose the following formulation for $1|r_j, s_j, B|C_{\max}$:

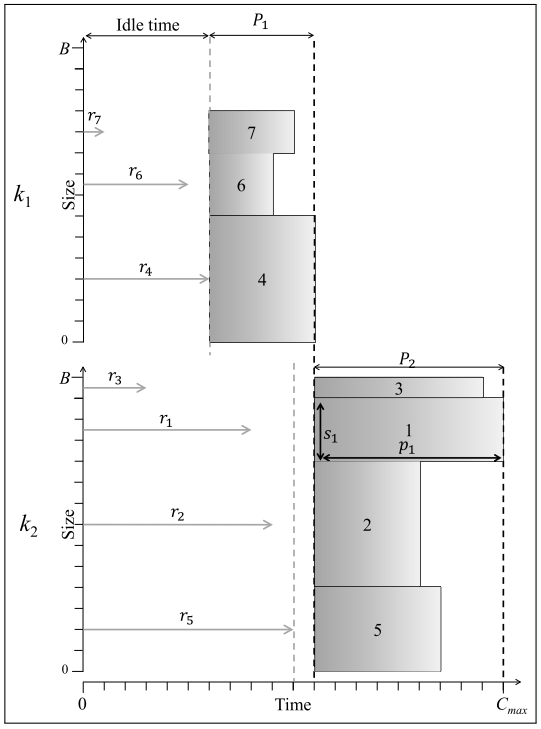
$$\begin{aligned}
(\text{MILP}_3^+) \quad & \min\ S_{n_K} + P_{n_K}, && && (49)\\
& (39),\ (40),\ (41),\ (42), && && (50)\\
& P_k \geq p_j x_{jk}, && \forall j \in J, \forall k \in K : j \leq k, && (51)\\
& P_k \geq 0, && \forall k \in K, && (52)\\
& S_k \geq r_k x_{kk}, && \forall k \in K, && (53)\\
& S_k \geq S_{k-1} + P_{k-1}, && \forall k \in K : k > 1, && (54)\\
& S_k \geq 0, && \forall k \in K. && (55)
\end{aligned}$$

The objective function (49) minimizes the makespan, given by the time required to finish processing the last batch. Constraints (51)-(52) determine the processing time of the batches. Constraints (53)-(55) determine when each batch starts to be processed. Constraints (39), (40), (41), and (42) are taken from (MILP_1^+).

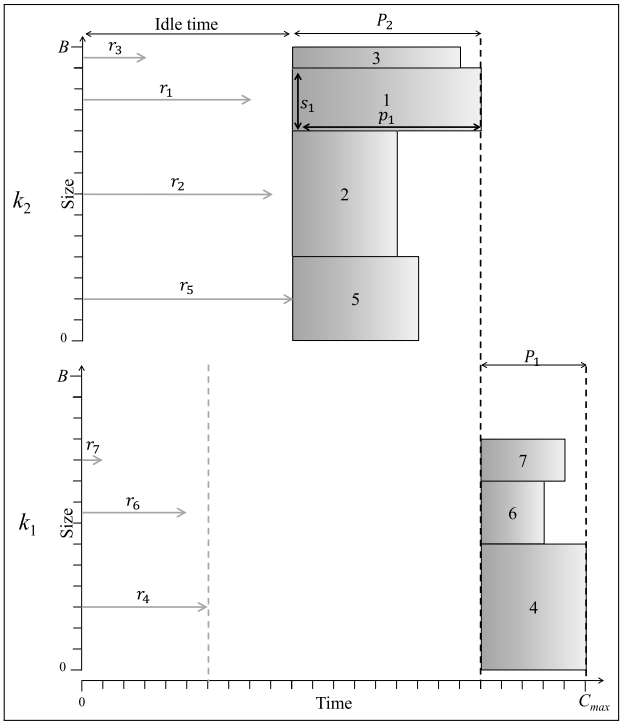
The following proposition certifies that our formulation is equivalent to formulation (MILP_3), concerning the optimal solution value.

Proposition 2. *The optimal makespan of problems (MILP$_3$) and (MILP$_3^+$) are the same.*

Proof. Let us consider w.l.o.g. that the indexes of jobs in J satisfy (48). Clearly, any feasible solution of (MILP_3^+) is also a feasible solution to (MILP_3) with the same objective function value.



(a) Solution for $1|r_j, s_j, B|C_{\max}$ with batch k_1 processed before k_2.



(b) Solution for $1|r_j, s_j, B|C_{\max}$ with batch k_2 processed before k_1.

Figure 3: An example of solution for $1|r_j, s_j, B|C_{\max}$.

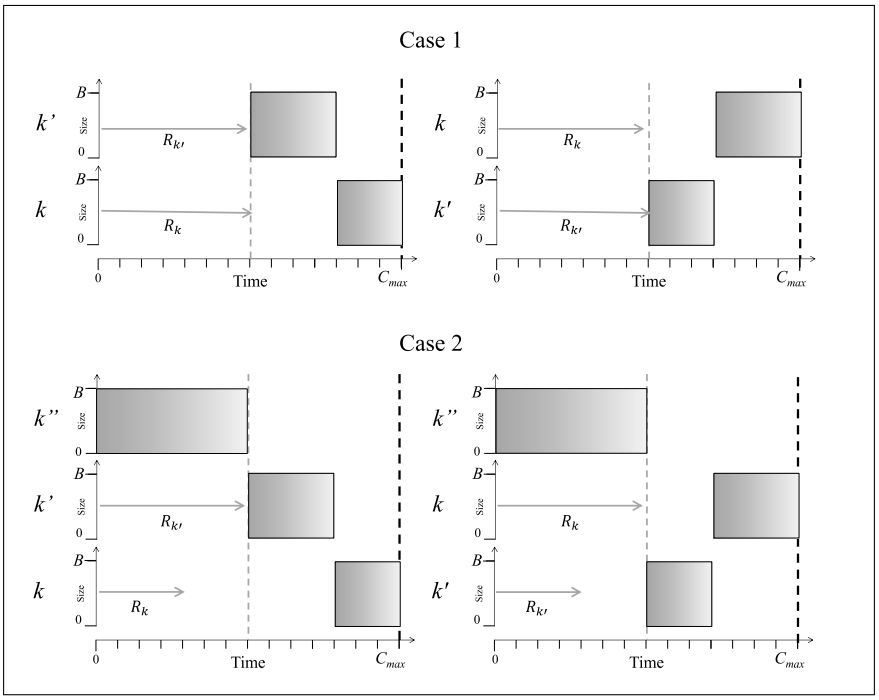


Figure 4: Cases where switching batches in the processing order does not modify the makespan.

Therefore, it suffices to show that given an optimal solution to (MILP$_3$), with objective function value $\bar{C}_{\max}$, there is a feasible solution to (MILP$_3^+$) also with objective function value $\bar{C}_{\max}$. For that, let us first reset, if necessary, the indexes of the batches on the given solution of (MILP$_3$), determining them as the largest index among the ones of all jobs assigned to it. The solution is now a feasible to solution to (MILP$_3^+$), with the batches designed as in the given feasible solution to (MILP$_3$), but possibly processed in a different order. Note that as the batches are now ordered by non-decreasing release times, the reordering cannot increase the idle time of the machine, and thus, cannot increase the makespan. Therefore the feasible solution to (MILP$_3^+$) also have the minimum makespan $C_{\max}$, as the objective function value. □

As we did for problem $P_m|s_j, B|C_{\max}$, we will now replace the variable x_{jkm} in (MILP$_4$) with the binary variables x_{jk} e y_{km}. By doing so, we can apply to $P_m|r_j, s_j, B|C_{\max}$, the same symmetry breaking strategy used for $1|r_j, s_j, B|C_{\max}$. Considering that the indexes of the jobs satisfy the relation (48), we propose the following formulation for $P_m|r_j, s_j, B|C_{\max}$.

$$
\begin{aligned}
(\text{MILP}_4^+)\quad & \min\ C_{\max}, && && (56)\\
& (39),\ (40),\ (41),\ (42), && && (57)\\
& x_{kk} \le \sum_{m\in M} y_{km}, && \forall k \in K, \forall m \in M, && (58)\\
& P_{km} \ge p_j(x_{jk} + y_{km} - 1), && \forall j \in J, \forall k \in K : j \le k, \forall m \in M, && (59)\\
& P_{km} \ge 0, && \forall k \in K, \forall m \in M, && (60)\\
& S_{km} \ge r_k y_{km}, && \forall k \in K, \forall m \in M, && (61)\\
& S_{km} \ge S_{(k-1)m} + P_{(k-1)m}, && \forall k \in K : k > 1, \forall m \in M, && (62)\\
& S_{km} \ge 0, && \forall k \in K, \forall m \in M, && (63)\\
& C_{\max} \ge S_{n_K m} + P_{n_K m}, && \forall m \in M, && (64)\\
& C_{\max} \ge 0. && && (65)
\end{aligned}
$$

The objective function (56) minimizes the makespan. Constraints (58) ensure that all batches used are assigned to a machine. Constraints (59)-(60) determine the processing time of each batch in each machine. Constraints (61)-(63) determine when each batch starts to be processed on the machine to which it is assigned. Constraints (64)-(65) determine the makespan as the time required to finish processing the last batch on all machines. Constraints (39), (40), (41), and (42) are taken from (MILP$_1^+$).

5 Computational results

In order to compare our formulations to the formulations proposed in the literature for the problems addressed in this paper, and also to evaluate the strength of the symmetry treatment that we propose, we performed an extensive numerical experiment, solving several instances of the problems with the MILP solver `CPLEX`, version 12.5. In all tests, `CPLEX` was configured to run in only one thread, to not benefit from the processor parallelism. The experiments with models (MILP$_1$) and (MILP$_1^+$) of problem $1|s_j, B|C_{\max}$ were performed on a computer with a `Intel Core 2` 2GHz processor, and 2GB of RAM. For all the other three problems, we used a computer with a 2.83 GHz `Intel Quad-Core Xeon X3360` processor, and 8GB of RAM. The computational time to solve each instance of all problems was limited in 1800 seconds. The set of test instances for problem $1|s_j, B|C_{\max}$ is the same one considered in [3], made available by the authors. For the other problems, the test instances were randomly generated following directions given in the literature, as specified next. The following statistics were considered in our analysis for all problems: the makespan corresponding to the best solution obtained by `CPLEX` ($C_{\max}$), the computational time of `CPLEX` in seconds ($T(s)$) (when `CPLEX` reaches the time limit of 1800 seconds on all instances of a given configuration, we represent the time by the symbol "-" in our tables), the time required by `CPLEX` to obtain the best solution, in seconds ($T_b(s)$), and the duality gap of `CPLEX` at the end of its execution (Gap).

All values presented in the tables are the average results computed over all instances of the same configuration, which is specifically explained in the remaining subsections. Therefore, note that it is possible to have the computational times of `CPLEX` for solving problems of a given configuration less than 1800 seconds while the gaps are non-zero. This happens when some of the instances in the group could be solved to optimality in 1800 seconds, and others could not.

In Table 1 we present all instance parameters utilized in this work. Each problem has own parameter settings, based on the literature review.

Table 1: Instance parameters for problems.

	$1\|s_j, B\|C_{\max}$	$P_m\|s_j, B\|C_{\max}$	$1\|r_j, s_j, B\|C_{\max}$	$P_m\|r_j, s_j, B\|C_{\max}$
Number of jobs:	n_J: 10, 20, 50, 100, 200, 300, 500	n_J: 10, 20, 50, 100	n_J: 10, 20, 50, 100, 200, 300, 500	n_J: 10, 20, 50, 100
Number of parallel machines:	-	n_M: 2, 4, 8	-	n_M: 2, 4, 8
Machine capacity:	10	10	40	40
Range for job processing time:	$p1$: [1, 10] $p2$: [1, 20]	$p1$: [1, 10] $p2$: [1, 20]	$p1$: [8, 48]	$p1$: [8,48]
Range for job size:	$s1$:[1, 10] $s2$:[2, 4] $s3$:[4, 8]	$s1$:[1, 10] $s2$:[2, 4] $s3$:[4, 8]	$s1$: [1, 15] $s2$: [15, 35]	$s1$: [1, 15] $s2$: [15, 35]
Range for job release time:	-	-	r1: [0, C]	r1: [0, C]

In Table 2, we illustrate the impact of our reformulations, presenting the number of variables and constraints for each model considered in our work, for an instance with 100 jobs.

5.1 Problem $1|s_j, B|C_{\max}$

The test instances for problem $1|s_j, B|C_{\max}$ were the same ones considered in [3], and were generated as suggested in [19]. To generate the instances, seven different number of jobs (n_J) were considered, as well as two different ranges for the processing times ($p1$ and $p2$) and three different ranges for the

Table 2: Impact on number of variables and constraints when $n_J = 100$ and $n_M = 4$.

	$1\|s_j, B\|C_{\max}$		$P_m\|s_j, B\|C_{\max}$		$1\|r_j, s_j, B\|C_{\max}$		$P_m\|r_j, s_j, B\|C_{\max}$	
	MILP_1	MILP_1^+	MILP_2	MILP_2^+	MILP_3	MILP_3^+	MILP_4	MILP_4^+
Binary variables	10100	5050	40000	5450	10000	5050	40000	5450
Continuous variables	100	0	401	1	200	200	401	801
Constraints	20200	5250	40204	5354	20399	10499	80104	36254

sizes of the jobs ($s1$, $s2$ and $s3$), in which they were randomly selected. Considering the parameters shown in Table 1, instances of $42(7 \times 2 \times 3)$ configurations were tested. For each configuration, 100 instances were generated, totalizing 4200 instances.

Tables 3-4 show average computational results for each group of 100 instances with each configuration, determined by the number of jobs, and by the ranges used to select the processing times and the sizes of the jobs. The first column shows the configuration of the instances, according to Table 1. The other columns show statistics analyzed for both models (MILP_1) and (MILP_1^+).

Table 3: Computational results for $1|s_j, B|C_{\max}$.

Instance		(MILP_1)				(MILP_1^+)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	C_{max}	$T(s)$	$T_b(s)$	Gap
10	p1s1	36.86	0.12	0.10	0.00	36.86	0.01	0.01	0.00
10	p1s2	20.38	0.06	0.06	0.00	20.38	0.02	0.02	0.00
10	p1s3	43.79	0.18	0.12	0.00	43.79	0.00	0.00	0.00
10	p2s1	67.62	0.11	0.10	0.00	67.62	0.01	0.01	0.00
10	p2s2	40.22	0.06	0.06	0.00	40.22	0.02	0.02	0.00
10	p2s3	81.05	0.15	0.12	0.00	81.05	0.00	0.00	0.00
20	p1s1	68.07	588.79	1.77	1.15	68.07	0.03	0.03	0.00
20	p1s2	37.13	1297.88	18.67	19.76	37.13	0.18	0.18	0.00
20	p1s3	83.69	478.03	1.99	0.93	83.69	0.01	0.01	0.00
20	p2s1	133.09	633.27	2.19	0.85	133.09	0.03	0.03	0.00
20	p2s2	72.88	1459.43	31.31	21.87	72.88	0.21	0.19	0.00
20	p2s3	159.11	741.51	2.48	1.56	159.11	0.01	0.01	0.00
50	p1s1	164.43	-	341.79	51.84	164.08	1.03	0.21	0.00
50	p1s2	88.37	-	472.65	69.35	87.39	415.40	3.88	0.23
50	p1s3	202.09	-	352.51	61.64	202.03	0.04	0.10	0.00
50	p2s1	315.43	-	362.54	55.00	314.57	0.61	0.27	0.00
50	p2s2	170.22	-	371.67	70.02	168.11	265.89	17.42	0.08
50	p2s3	384.40	-	397.58	66.59	384.13	0.04	0.10	0.00
100	p1s1	346.95	-	283.16	89.77	318.99	82.41	1.50	0.02
100	p1s2	208.96	-	292.26	88.39	170.59	1630.65	96.89	1.26
100	p1s3	414.51	-	315.37	91.13	396.96	0.10	0.10	0.00
100	p2s1	670.27	-	314.30	90.40	610.64	51.01	9.16	0.01
100	p2s2	417.73	-	351.37	90.20	326.26	1659.79	235.46	0.97
100	p2s3	805.43	-	270.74	91.92	766.91	0.11	0.11	0.00

The comparative tests clearly show that formulation (MILP_1^+) is superior to (MILP_1). For instances with 50 jobs or more, formulation (MILP_1) does not prove optimality for any instance, while (MILP_1^+) shows better results in less computational time. Additionally, the duality gaps shown for (MILP_1) reveal the difficulty in obtaining good lower bounds. This difficulty is mitigated by the use of (MILP_1^+), which obtains better lower bounds even when optimality is not proven.

Through the computational times shown for both formulations, we conclude that instances with configuration $s2$ are in general, more time consuming, and may be considered more difficult than instances with configuration $s1$ for both formulations. The reason for this is the small sizes of the jobs when compared to the machine capacity, which allows more combinations of jobs that are assigned to a batch. Consequently, the feasible set of instances with this configuration tend to be larger than for instances with configuration $s1$, leading to more time needed for the solver to prove optimality. For these instances the solver usually find the best solution much faster than it proves optimality.

Table 4: Computational results for $1|s_j, B|C_{\max}$ - 200-500 jobs

Instance		(MILP_1^+)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
200	p1s1	629.43	191.47	14.97	0.02
200	p1s2	333.76	1634.52	280.11	0.96
200	p1s3	786.19	21.40	0.43	0.00
200	p2s1	1197.48	262.14	33.48	0.02
200	p2s2	638.63	1635.79	377.39	0.95
200	p2s3	1505.11	0.58	0.47	0.00
300	p1s1	928.68	557.85	65.79	0.06
300	p1s2	497.01	1499.80	427.03	0.58
300	p1s3	1174.46	36.50	1.48	0.00
300	p2s1	1793.54	530.28	119.89	0.03
300	p2s2	966.69	1532.10	418.01	0.83
300	p2s3	2247.39	24.44	1.66	0.00
500	p1s1	1544.36	1171.28	247.34	0.10
500	p1s2	835.00	1637.21	223.91	0.68
500	p1s3	1949.76	64.22	6.11	0.00
500	p2s1	2964.93	1277.59	337.54	0.09
500	p2s2	1599.44	1750.57	284.53	1.12
500	p2s3	3701.79	67.96	5.94	0.00

5.2 Problem $P_m|s_j, B|C_{\max}$

The test instances for problem $P_m|s_j, B|C_{\max}$ were randomly generated following the procedure described in [2], but considering different numbers of jobs. As it was done for $1|s_j, B|C_{\max}$ two different ranges for the processing times and three different ranges for the sizes are set for the jobs, and their values are randomly selected from the ranges. The $72(4 \times 3 \times 2 \times 3)$ configurations of the instances generated are shown in Table 1. For each configuration, we generated 100 random instances, summing up to 7200 test instances.

Table 5 show average computational results for each group of 100 instances with each configuration tested. The first and second columns present the configuration of the instances, according to Table 1. The other columns show the statistics analyzed for both models (MILP_2) and (MILP_2^+).

Table 5: Computational results for $P_m|s_j, B|C_{\max}$ - 2-8 parallel machines.

Instance		(MILP_2)				(MILP_2^+)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
2 parallel machines									
10	p1s1	18.76	0.13	0.12	0.00	18.76	0.01	0.01	0.00
10	p1s2	11.03	0.05	0.05	0.00	11.03	0.02	0.02	0.00
10	p1s3	22.13	0.19	0.15	0.00	22.13	0.01	0.01	0.00
10	p2s1	34.50	0.12	0.11	0.00	34.50	0.01	0.01	0.00
10	p2s2	21.71	0.05	0.05	0.00	21.71	0.02	0.02	0.00
10	p2s3	40.87	0.17	0.14	0.00	40.87	0.01	0.01	0.00
20	p1s1	34.27	1308.41	45.82	5.54	34.27	0.03	0.03	0.00
20	p1s2	18.83	884.08	21.46	8.16	18.83	0.11	0.11	0.00
20	p1s3	42.13	1412.74	27.48	6.27	42.13	0.02	0.02	0.00
20	p2s1	66.79	1287.70	27.99	4.35	66.79	0.03	0.03	0.00
20	p2s2	36.87	651.70	24.56	7.05	36.87	0.15	0.15	0.00
20	p2s3	79.82	1395.83	46.57	5.60	79.82	0.02	0.02	0.00
50	p1s1	83.07	-	859.53	58.36	82.30	2.48	0.24	0.00
50	p1s2	46.56	-	1665.10	59.68	43.94	529.33	5.26	0.52
50	p1s3	101.74	-	700.81	60.69	101.30	0.02	0.02	0.00
50	p2s1	159.08	-	1069.49	61.30	157.52	5.12	0.33	0.00
50	p2s2	88.96	-	1733.41	62.44	84.32	478.37	15.40	0.19
50	p2s3	192.95	-	985.71	64.02	192.34	0.03	0.03	0.00

Continued on next page

Table 5 – *Continued from previous page*

Instance		(MILP$_2$)				(MILP$_2^+$)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
100	p1s1	171.60	-	1515.19	87.71	159.78	192.10	2.47	0.07
100	p1s2	98.19	-	1550.56	86.49	85.56	1743.59	47.26	1.73
100	p1s3	206.66	-	1475.05	86.52	198.75	0.15	0.09	0.00
100	p2s1	328.38	-	1630.23	89.47	305.58	84.36	5.71	0.02
100	p2s2	188.69	-	1562.67	88.60	163.39	1770.79	280.17	1.21
100	p2s3	398.94	-	1514.87	89.28	383.73	0.20	0.11	0.00
				4 parallel machines					
10	p1s1	10.87	0.16	0.14	0.00	10.87	0.02	0.02	0.00
10	p1s2	9.49	0.10	0.09	0.00	9.49	0.01	0.01	0.00
10	p1s3	12.18	0.25	0.22	0.00	12.18	0.02	0.02	0.00
10	p2s1	20.26	0.16	0.12	0.00	20.26	0.02	0.02	0.00
10	p2s2	18.68	0.11	0.10	0.00	18.68	0.01	0.01	0.00
10	p2s3	22.67	0.23	0.21	0.00	22.67	0.02	0.02	0.00
20	p1s1	17.47	1316.19	87.17	8.11	17.47	0.05	0.05	0.00
20	p1s2	10.43	56.14	1.59	0.49	10.43	0.32	0.32	0.00
20	p1s3	21.29	1629.93	95.67	11.78	21.29	0.03	0.03	0.00
20	p2s1	33.95	1122.49	73.44	5.29	33.95	0.07	0.07	0.00
20	p2s2	20.51	92.62	7.26	0.64	20.51	0.35	0.35	0.00
20	p2s3	40.21	1731.24	145.29	12.27	40.21	0.05	0.05	0.00
50	p1s1	42.69	-	1077.53	70.03	41.43	2.54	0.17	0.00
50	p1s2	23.76	-	1674.71	58.83	22.18	269.31	7.31	0.56
50	p1s3	51.85	-	995.45	71.91	50.90	0.05	0.05	0.00
50	p2s1	81.23	-	1214.43	71.19	78.97	0.90	0.62	0.00
50	p2s2	45.55	-	1715.65	57.37	42.38	283.70	26.62	0.19
50	p2s3	97.89	-	1234.96	73.39	96.40	0.07	0.07	0.00
100	p1s1	93.06	-	1642.24	93.44	80.09	82.33	2.28	0.05
100	p1s2	50.26	-	1339.98	81.80	43.06	1409.33	52.28	1.67
100	p1s3	110.60	-	1614.86	92.94	99.64	0.55	0.17	0.00
100	p2s1	177.17	-	1707.58	93.54	153.03	51.98	6.25	0.02
100	p2s2	96.18	-	1450.97	86.63	82.03	1679.11	279.71	1.32
100	p2s3	213.47	-	1685.64	93.38	192.11	0.52	0.50	0.00
				8 parallel machines					
10	p1s1	9.54	0.23	0.10	0.00	9.54	0.01	0.01	0.00
10	p1s2	9.49	0.25	0.10	0.00	9.49	0.01	0.01	0.00
10	p1s3	9.42	0.33	0.13	0.00	9.42	0.01	0.01	0.00
10	p2s1	18.55	0.21	0.09	0.00	18.55	0.01	0.01	0.00
10	p2s2	18.68	0.24	0.10	0.00	18.68	0.01	0.01	0.00
10	p2s3	18.27	0.34	0.12	0.00	18.27	0.01	0.01	0.00
20	p1s1	10.51	276.62	15.22	2.44	10.51	0.09	0.09	0.00
20	p1s2	9.81	2.76	0.23	0.00	9.81	0.07	0.07	0.00
20	p1s3	11.61	760.27	28.35	7.24	11.60	0.15	0.15	0.00
20	p2s1	20.76	328.01	14.35	3.34	20.76	0.13	0.13	0.00
20	p2s2	19.52	2.80	0.23	0.00	19.52	0.08	0.08	0.00
20	p2s3	22.31	958.29	37.31	8.28	22.30	0.26	0.26	0.00
50	p1s1	22.30	-	1219.28	55.90	20.96	2.99	0.24	0.00
50	p1s2	12.83	1783.20	1505.90	27.02	11.77	850.12	6.42	4.12
50	p1s3	26.78	-	1417.79	62.67	25.71	0.10	0.10	0.00
50	p2s1	42.41	-	1495.51	55.68	39.72	1.25	1.19	0.00
50	p2s2	24.41	1775.39	1617.85	28.90	22.46	1198.77	75.59	3.91
50	p2s3	50.33	-	1546.21	61.21	48.45	0.17	0.17	0.00
100	p1s1	59.57	-	1304.89	96.84	40.34	51.78	3.32	0.05
100	p1s2	28.80	-	1171.47	84.72	21.82	872.63	62.93	2.04
100	p1s3	69.72	-	1164.84	98.03	50.07	0.22	0.22	0.00
100	p2s1	123.38	-	1285.47	97.70	76.81	59.45	18.86	0.01
100	p2s2	57.82	-	1293.37	94.95	41.34	1251.21	230.46	1.45
100	p2s3	139.99	-	1051.10	98.19	96.34	0.81	0.81	0.00

The comparative tests show that formulation (MILP$_2^+$) obtain in general solutions that are better or at least equivalent to the solutions obtained by (MILP$_2$). It also runs in less computational time for all instances considered. We note that both formulations run in small computational times for

instances with 10 jobs, and optimality is proven with both formulations for all instances in this group. On the other side, for instances with 20 jobs, (MILP$_2^+$) overcomes (MILP$_2$), proving optimality for these instances in reduced computational time. Formulation (MILP$_2$) reveals difficulty in solving instances with 20 jobs. Even being able to prove optimality for most of them, the computational times and the duality gap for some instances are large, particularly when less parallel machines are used. It is possible to see that (MILP$_2$) does not prove optimality for all instances with 20 jobs, with configurations $p_1 s_3$ and $p_2 s_3$.

Instances of type $s2$ tend to consume more computational effort than instances of type $s1$, and can be considered more difficult. This behavior can also be observed in the computational experiments of the previous section. This difference can be explained by the small size of the jobs, which increases the number possible combinations of jobs assigned to each batch, generating a larger feasible set for the problem. It is also observed that the increase in the number of parallel machines results in a decrease of the computational times.

5.3 Problem $1|r_j, s_j, B|C_{\max}$

The set of instances considered for problem $1|r_j, s_j, B|C_{\max}$ was the same used in [24], and was generated according to the methodology proposed in [6]. For each job j, a processing time p_j, a release time r_j, and a size s_j are randomly selected as described in Table 1. In total, 280 instances were generated, 20 for each of the 14 combinations of number and size of jobs.

The following steps were considered to generate the release times of the jobs:

1. For each instance, the size and processing time of each job, are randomly selected in the intervals presented in Table 1.
2. A lower bound C on the optimal makespan for each instance is computed. This bound does not consider the release times of the jobs, which have not been defined yet. For this computation, a simple batch-first-fit heuristic, proposed in [22], is applied generating a feasible solution for the problem with no release times.
3. Finally, the job release times for each instance, are randomly generated in the interval [0, C], where C is the value of the lower bound found in the previous step.

Table 6 presents the average computational results for the 20 instances tested with each configuration. The two first columns show the instances configuration, now specified by the pair of parameters n_J, si, for $i = 1, 2$, according to Table 1. The other columns show the statistics analyzed for both formulations (MILP$_3$) and (MILP$_3^+$).

Table 6: Computational results for $1|r_j, s_j, B|C_{\max}$.

Instance		(MILP$_3$)				(MILP$_3^+$)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
10	$s1$	117.80	0.16	0.09	0.00	117.80	0.02	0.02	0.00
10	$s2$	316.95	0.57	0.20	0.00	316.95	0.01	0.00	0.00
20	$s1$	193.80	192.34	27.78	0.03	193.80	0.33	0.32	0.00
20	$s2$	560.70	270.13	7.97	0.44	560.70	0.01	0.01	0.00
50	$s1$	396.50	-	1674.20	40.64	389.45	456.99	129.53	0.63
50	$s2$	1351.80	-	1725.32	75.01	1298.55	0.06	0.06	0.00
100	$s1$	920.05	-	1744.30	94.50	760.45	1744.24	327.94	5.51
100	$s2$	3368.70	-	1640.57	94.90	2578.35	0.39	0.36	0.00
200	$s1$	2884.10	-	177.67	98.33	1576.75	-	1761.36	15.38
200	$s2$	9257.15	-	82.61	99.40	5049.35	1.19	1.17	0.00
300	$s1$	4355.70	-	1785.86	99.94	2526.05	-	1388.04	21.49
300	$s2$	13707.75	-	1754.98	99.86	7483.25	1.99	1.93	0.00
500	$s1$	7152.60	-	-	100.00	5805.50	-	1179.36	43.76
500	$s2$	23320.85	-	-	100.00	12589.80	3.50	3.43	0.00

Both formulations (MILP$_3$) and (MILP$_3^+$) are efficient for solving instances of reduced size and have proved optimality for all instances with 10 jobs. However, the remaining results show that

(MILP_3^+) overcomes (MILP_3), both in computational time and in the quality of the solutions obtained. Formulation (MILP_3) cannot prove optimality of any instance with more than 20 jobs, while (MILP_3^+) proves optimality for all instances with 10 and 20 jobs in no more than 0.32 seconds. Considering instances with 50 and 100 jobs, (MILP_3^+) continues to outperform (MILP_3) in every respect.

Table 6 also shows that instances of type $s1$ are more difficult than than instances of type $s2$ for both formulations, which was already verified in previous computational experiments. As before, the reduced size of the jobs generates a larger number of combinations in the batch configuration. Formulation (MILP_3^+) finds all optimal solutions for instances of type $s2$, with average computational time less than 3.5 seconds. For instances of type $s1$, with more than 50 jobs, (MILP_3^+) cannot prove optimality for all instances, but obtains much better solutions and lower bounds than (MILP_3).

5.4 Problem $P_m|r_j, s_j, B|C_{\max}$

For the computational experiments on problem $P_m|r_j, s_j, B|C_{\max}$, we used the same set of instances used for problem $1|r_j, s_j, B|C_{\max}$. In addition, three new categories were included corresponding to the numbers of parallel machines: 2, 4 and 8 machines, and each instance was tested for the three different numbers of parallel machines. The sizes, processing times and release times of the jobs were randomly selected in the ranges shown in Table 1. According to the table, 8 instance configurations were generated, defined by the combination of four different number of jobs, and two ranges for the job sizes. For each configuration, 20 instances were generated, summing up to 160 instances. Each instance was tested for the three numbers of parallel machine, leading to 480 tests performed.

The release times generation following the same procedure described in the last section 5.3, for the problem $1|r_j, s_j, B|C_{\max}$.

Table 7 show average computational results for each group of 100 instances with each configuration. The two first columns show the configuration of the instance, according to Table 1. The other columns show the statistics analyzed for both formulations (MILP_4) and (MILP_4^+).

The results shown in this subsection show that, in general, (MILP_4^+) obtain better solutions than (MILP_4) in less computational time. (MILP_4^+) proves optimality for all instances with up to 50 jobs, while (MILP_4) can only prove optimality for all instances with up to 20 jobs. Furthermore, even for these instances, it takes more time to run than (MILP_4^+).

For instances with 50 jobs, the superiority of (MILP_4^+) over (MILP_4) is even clearer. It can prove optimality for all instances in reduced times, while (MILP_4) has difficulty and does not prove optimality for the majority of the instances. The computational times and the duality gaps for (MILP_4) are big, especially when compared to (MILP_4^+), which solve the instances in less than 6.46 seconds on average. Following the pattern of the previous tests, instances of type $s2$ consume more computational effort and are more difficult than those of type $s1$.

6 Conclusions

In this paper, we address four different versions of batch scheduling problems, which have been identified in the literature as suited models for problems that appear in reliability tests in the semiconductor industry. The economic importance of the problems have motivated the investigation of good solution approaches for them, and their NP-hardness have led the majority of this research to focus on heuristic approaches. We show in this paper that applying good MILP formulations for these scheduling problems we can go a step further in the exact resolution of applied problems, having presented optimal solutions for test instances with sizes never considered in the literature by exact methods. Even for instances which we could not solve to optimality in our time limit of 1800 seconds, we were able to present much better average results with the formulations proposed than the ones obtained with formulations previously presented in the literature.

The enhancement in the models was mainly based on the idea of eliminating symmetric solutions from the feasible sets of the problems. The development of symmetry breaking cuts is widely pursued

Table 7: Computational results for $P_m|r_j, s_j, B|C_{\max}$ - 2-8 parallel machines.

Instance		(MILP$_4$)				(MILP$_4^+$)			
jobs	type	$C_{\max}$	$T(s)$	$T_b(s)$	Gap	$C_{\max}$	$T(s)$	$T_b(s)$	Gap
2 parallel machines									
10	$s1$	106.80	0.03	0.03	0.00	106.80	0.03	0.03	0.00
10	$s2$	281.15	0.06	0.04	0.00	281.15	0.01	0.01	0.00
20	$s1$	177.05	1.28	0.48	0.00	177.05	0.13	0.11	0.00
20	$s2$	521.25	10.28	1.09	0.00	521.25	0.02	0.02	0.00
50	$s1$	362.15	706.72	47.89	0.74	362.15	2.29	2.01	0.00
50	$s2$	1254.05	1653.55	182.58	40.69	1253.40	0.06	0.06	0.00
100	$s1$	707.40	-	1659.15	94.79	691.00	53.53	28.30	0.00
100	$s2$	2666.95	-	1558.00	90.79	2501.45	0.18	0.18	0.00
4 parallel machines									
10	$s1$	106.60	0.02	0.02	0.00	106.60	0.03	0.03	0.00
10	$s2$	278.10	0.03	0.03	0.00	278.10	0.01	0.01	0.00
20	$s1$	176.70	0.67	0.15	0.00	176.70	0.28	0.13	0.00
20	$s2$	518.35	0.89	0.23	0.00	518.35	0.03	0.03	0.00
50	$s1$	361.55	367.34	10.23	0.00	361.55	2.54	1.11	0.00
50	$s2$	1251.40	1624.65	12.46	8.23	1251.40	0.19	0.13	0.00
100	$s1$	690.05	-	451.21	73.25	690.05	28.76	9.31	0.00
100	$s2$	2498.55	1636.68	599.11	74.42	2498.55	1.32	0.66	0.00
8 parallel machines									
10	$s1$	106.60	0.09	0.08	0.00	106.60	0.07	0.06	0.00
10	$s2$	278.10	0.04	0.03	0.00	278.10	0.02	0.02	0.00
20	$s1$	176.70	0.47	0.13	0.00	176.70	1.39	0.23	0.00
20	$s2$	518.35	0.99	0.13	0.00	518.35	0.04	0.04	0.00
50	$s1$	361.55	336.81	7.95	0.01	361.55	6.46	1.92	0.00
50	$s2$	1251.40	950.79	7.75	1.45	1251.40	0.47	0.25	0.00
100	$s1$	690.05	1653.79	108.07	30.62	690.05	45.43	12.39	0.00
100	$s2$	2498.55	1657.19	322.27	83.48	2498.55	4.13	1.46	0.00

in the MILP literature and some general approaches can certainly be applied to scheduling problems. Nevertheless, using some well-known properties of the optimal solutions of the problems addressed, we propose a specific indexing of the jobs to be processed, for each version of the problem. The indexing not only allows the reduction the feasible sets by eliminating symmetric solutions but also significantly reduces the number of variables and constraints in the models, when compared to the ones in the literature, leading to simplified and stronger formulations. Finally, the referred properties of optimal solutions are the backbone to the proof of correctness of the models presented in this paper.

As future research, we would like to investigate if the good performance of the models presented can be replicated when symmetry breaking constraints are applied to other problems in the vast area of scheduling applications as, for example, the problem of scheduling a batch processing machine with incompatible job families. It would be also interesting to investigate the use of symmetry breaking constraints in a branch-and-price algorithm applied to the problems addressed in this work.

Acknowledgments

Renan Spencer Trindade was supported by a Ph.D. scholarship from the The Brazilian National Council for Scientific and Technological Development (CNPq) [grant number 142205/2014-1]; Marcia Helena Costa Fampa was partially supported by CNPq [grant number 303898/2016-0].

References

[1] Al-Salamah, M. 2015. "Constrained binary artificial bee colony to minimize the makespan for single machine batch processing with non-identical job sizes." *Applied Soft Computing* 29: 379—385.

[2] Chang, P.-Y. and Damodaran *, P. and Melouk, S. 2004. "Minimizing makespan on parallel batch processing machines." *International Journal of Production Research* 42(19): 4211–4220.

[3] Chen, H., Du, B., and Huang, G. Q. 2011. "Scheduling a batch processing machine with non-identical job sizes: a clustering perspective." *International Journal of Production Research* 49(19), 5755–5778.

[4] Cheng, B., Yang, S., Hu, X., and Chen, B. 2012. "Minimizing makespan and total completion time for parallel batch processing machines with non-identical job sizes." *Applied Mathematical Modelling* 36(7): 3161-–3167.

[5] Cheng, B., Wang, Q., Yang, S., and Hu, X. 2013. "An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes." *Applied Soft Computing* 13(2): 765-–772.

[6] Chou, F.-D., Chang, P.-C., and Wang, H.-M. 2005. "A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem." *The International Journal of Advanced Manufacturing Technology* 31(3–4): 350-–359.

[7] Chung, S. H., Tai, Y. T., and Pearn, W. L. 2009. "Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes." *International Journal of Production Research* 47(18): 5109—5128.

[8] Damodaran, P., Ghrayeb, O., and Guttikonda, M. C. 2013. "GRASP to minimize makespan for a capacitated batch-processing machine." *The International Journal of Advanced Manufacturing Technology* 68(1–4): 407—414.

[9] Damodaran, P., Kumar Manjeshwar, P., and Srihari, K. 2006. "Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms." *International Journal of Production Economics* 103(2): 882-–891.

[10] Damodaran, P. , N. S. Hirani , and M. C. Velez-Gallego. 2009. "Scheduling Identical Parallel Batch Processing Machines to Minimise Makespan Using Genetic Algorithms." *European Journal of Industrial Engineering* 3(2): 187–206.

[11] Damodaran, P., Velez-Gallego, M. C., and Maya, J. 2009. "A GRASP approach for makespan minimization on parallel batch processing machines. Journal of Intelligent Manufacturing." *Journal of Intelligent Manufacturing* 22(5): 767—777.

[12] Ghazvini, F. J., and Dupont, L. 1998. "Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes." *International Journal of Production Economics* 55: 273-–280.

[13] Jia, Z., and Leung, J. Y.-T. 2015. "A meta-heuristic to minimize makespan for parallel batch machines with arbitrary job sizes." *European Journal of Operational Research* 240(3): 649—665.

[14] Kashan, a. H., Karimi, B., and Jolai, F. 2006. "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes." *International Journal of Production Research* 44(12): 2337-–2360.

[15] Kashan, A. H., Karimi, B., and Jenabi, M. 2008. "A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes." *Computers & Operations Research* 35(4): 1084—1098.

[16] Köhler, V., Fampa, M., and Araújo, O. 2012. "Mixed-Integer Linear Programming Formulations for the Software Clustering Problem." *Computational Optimization and Applications*, 55(1): 113-–135.

[17] Lee, Y. H., and Lee, Y. H. 2013. "Minimising makespan heuristics for scheduling a single batch machine processing machine with non-identical job sizes." *International Journal of Production Research* 51(12): 3488-–3500.

[18] Margot, F. 2010. "Symmetry in Integer Linear Programming" *50 Years of Integer Programming* 1958–2008, 1—40, edited by M. Jünger, Th. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi and L. A. Wolsey. Berlin, Heidelberg: Springer.

[19] Melouk, S., Damodaran, P., and Chang, P.-Y. 2004. "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing." *International Journal of Production Economics* 87(2): 141-–147.

[20] Meng, Y., and Tang, L. 2010. "A tabu search heuristic to solve the scheduling problem for a batch-processing machine with non-identical job sizes." *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)* Vol. 3, 1703-–1707.

[21] Rafiee Parsa, N., Karimi, B., and Kashan, A. H. 2010 "A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes." *Computers & Operations Research* 37(10): 1720-–1730.

[22] Uzsoy, R. 1994. "Scheduling a single batch processing machine with non-identical job sizes." *International Journal of Production Research* 32(7): 1615—1635.

[23] Vélez-Gallego, M. C. 2009. "Algorithms for scheduling parallel batch processing machines with non-identical job ready times." Ph.D. thesis, Florida International University, Florida, United States.

[24] Xu, R., Chen, H., and Li, X. 2012. "Makespan minimization on single batch-processing machine via ant colony optimization." *Computers & Operations Research* 39(3): 582-–593.

[25] Zhang, G., Cai, X., Lee, C.-Y., and Wong, C. 2001. "Minimizing makespan on a single batch processing machine with nonidentical job sizes." *Naval Research Logistics* 48(3): 226-–240.

[26] Zhou, S., Chen, H., Xu, R., and Li, X. 2014. "Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes." *International Journal of Production Research* 52(8): 2258-–2274.