Arc-Flow Approach for Parallel Batch Processing Machine Scheduling with Non-identical Job Sizes

Renan Spencer Trindade, Olinto de Araújo, Marcia Fampa

Arc-flow approach for parallel batch processing machine scheduling with non-identical job sizes

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

[1] *LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France.*
rst@lix.polytechnique.fr
[2] *CTISM, Universidade Federal de Santa Maria, RS, Brazil.*
olinto@ctism.ufsm.br
[3] *IM, PESC/COPPE, Universidade Federal do Rio de Janeiro, RJ, Brazil.*
fampa@cos.ufrj.br

Abstract

Problems of minimizing makespan in scheduling batch processing machines are widely exploited by academic literature, mainly motivated by burn-in tests in the semiconductor industry. The problem addressed in this work consists of grouping jobs into batches and scheduling them in parallel machines. The jobs have non-identical size and processing times. The total size of the batch cannot exceed the capacity of the machine. The processing time of each batch will be equal to the longest processing time among all the jobs assigned to it. This paper proposes an arc-flow based model for minimizing makespan on parallel processing machines $P_m|s_j, B|C_{max}$. The mathematical model is solved using CPLEX, and computational results show that the proposed models have a better performance than other models in the literature.

Keywords: Parallel batch processing machine; Scheduling; Makespan; Arc-flow

1 Introduction

Scheduling is a widely used decision-making process in resource allocation and allows optimization in most production systems, information processing, transport, and distribution configurations, and several other real-world environments. This paper focuses on scheduling problems in Batch Processing Machines (BPM), that 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 BPM, the version 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

*Partially supported by a Ph.D. scholarship from the Brazilian National Council for Scientific and Technological Development (CNPq) [grant number 142205/2014-1] and by CNPq grant 303898/2016-0.

This is an Accepted Manuscript version of the following article. Trindade R.S., de Araújo O.C.B., Fampa M. (2020) Arc-Flow Approach for Parallel Batch Processing Machine Scheduling with Non-identical Job Sizes. In: Baïou M., Gendron B., Günlük O., Mahjoub A.R. (eds) Combinatorial Optimization. ISCO 2020. Lecture Notes in Computer Science, vol 12176. Springer, Cham. The final authenticated publication is available online at DOI: 10.1007/978-3-030-53262-8_15.

processing machine. The jobs need to be placed on a tray, respecting the capacity of the machine. The burn-in tests are a bottleneck in final testing operations, and the efficient scheduling of these operations aims to maximize productivity. The processing time to test an electronic circuit can reach up to 120 hours in a constant temperature around 120°C, as presented in [13]. On tests reported in [19] and [7], a liquid crystal display usually takes 6 hours to complete the reliability test, which reinforces the importance of an efficient scheduling.

The research on BPM is recent, compared to the history of the semiconductor manufacturing, and consists of grouping the jobs into batches. The publication [18] reviews the research done on scheduling models considering batch processing machines. A survey related to BPM problems research found in [15], analyzing publications between 1986 and 2004 (part of 2004 only). Another survey that focus on BPM problems published in [16] and reveals that p-batching is much more important in semiconductor manufacturing comparing with s-batching.

This paper considers $P_m|s_j, B|C_{\max}$ problem. In the literature, the works that address it are mostly extensions of the works published for the single machine version of the problem. In [3], the simulated annealing meta-heuristic is applied, and an Mixed Integer Linear Programming (MILP) formulation is presented for the problem. This work also proves the NP-hard complexity of the problem, and shows results for instances with up to 50 jobs. In [11], a hybrid genetic algorithm is used to compute solutions for instances with up to 100 jobs, considering 2 and 4 parallel machines. In [8] a new application of the genetic algorithm is proposed, which solves instances with up to 100 jobs, also on 2 and 4 parallel machines. In [6] an approximation algorithm is presented for the problem, with the approximation factor of 2. Finally, two other works that apply meta-heuristics ([5] and [10]), 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 [10], instances are solved with up to 100 jobs, on 2, 3, and 4 parallel machines. In [21] and [20], the authors propose a new formulation focused on symmetry breaking constraints.

We propose an arc-flow formulation for problem $P_m|s_j, B|C_{\max}$. The paper is organized as follows: In Section 2, we introduce problem $P_m|s_j, B|C_{\max}$ and present two formulations from the literature. In Section 3, we present an arc-flow based formulation for the problem. In Section 4, we discuss our numerical experiments comparing the arc-flow formulation to formulations from the literature. In Section 4, we present some concluding remarks and discuss future work.

2 Problem definition

The problem 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 assigned to a specific machine $M := \{1, \ldots, n_M\}$. All machines are identical, and each one has its own processing time, defined by the time of the last batch processed on the machine. 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 is 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 machine.

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

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

$$P_{km} : \text{time to process batch } k \text{ in machine } m. \quad (2)$$

$$C_{max} : \text{the makespan.} \quad (3)$$

In [3] the following MILP formulation is proposed for $P_m|s_j, B|C_{\max}$:

$$\text{(MILP)} \quad \min\ C_{\max}, \tag{4}$$

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

$$\sum_{j\in J}\sum_{m\in M} s_j x_{jkm} \leq B, \qquad \forall k \in K, \tag{6}$$

$$P_{km} \geq p_j x_{jkm}, \qquad \forall j \in J, \forall k \in K, \forall m \in M, \tag{7}$$

$$C_{\max} \geq \sum_{k\in K} P_{km}, \qquad \forall m \in M, \tag{8}$$

$$x_{jkm} \in \{0,1\}, \qquad \forall j \in J, \forall k \in K, \forall m \in M. \tag{9}$$

The objective function (4) minimizes the makespan. Constraints (5) and (6) ensure that each job is assigned to a single batch and a single machine, respecting the capacity of the machine. Constraints (7) determine the processing time of batch k in machine m. Constraints (8) determine the makespan, which is given by the longest sum of the processing times of all batches, among all machines. Note that formulation (MILP) 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. Note that constraints (6) take into account the fact that, although we have batches indexed by a given k, corresponding to all machines, a job can only be assigned to one of them, because of constraints (5). Therefore, a job j is only assigned to a unique pair (k, m).

(MILP) can be considered highly symmetrical concerning the order in which the batches are scheduled in each one of the parallel machines. This is because the same solution can be represented in different ways, just by changing the sequence order of the batches. In [21], and [20] the symmetry mentioned above is considered and a symmetry breaking procedure is used. At first, the variables x_{jkm} are replaced by two binary variables x_{jk}, which determine only the design of the batches, and the binary variables y_{km}, which determine whether or not batch k is processed in machine m. This replacement significantly reduces the number of binary variables. Furthermore, [21] presents a new formulation for the problem, where symmetric solutions are eliminated from the feasible set of (MILP), with the following approach. Firstly, the indexes of the jobs are defined by ordering them by their processing times. More specifically, it is considered that $p_1 \leq p_2 \leq \ldots \leq p_{n_J}$. Secondly, it is determined that batch k can only be used if job k is assigned to it, for all $k \in K$. Thirdly, it is determined that job j can only be assigned to batch k if $j \leq k$. Considering the above, the following formulation for $1|s_j, B|C_{\max}$ is proposed in [21]:

$$\text{(MILP}^+\text{)} \quad \min\ C_{\max}, \tag{10}$$

$$\sum_{k\in K:k\geq j} x_{jk} = 1, \qquad \forall j \in J, \tag{11}$$

$$\sum_{j\in J:j\leq k} s_j x_{jk} \leq B x_{kk}, \qquad \forall k \in K, \tag{12}$$

$$x_{jk} \leq x_{kk}, \qquad \forall j \in J, \forall k \in K, \tag{13}$$

$$x_{kk} \leq \sum_{m\in M} y_{km}, \qquad \forall k \in K, \tag{14}$$

$$C_m \geq \sum_{k\in K} p_k y_{km}. \qquad \forall m \in M, \tag{15}$$

$$C_{\max} \geq C_m \qquad \forall m \in M, \tag{16}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in J, \forall k \in K : j \leq k. \tag{17}$$

The objective function (10) minimizes the makespan given by the latest time to finish processing all batches in all machines. Constraints (11) determine that each job j is assigned to a single batch k, such that $k \geq j$. Constraints (12) 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 (13) are redundant together with (12), but are included to strengthen the linear relaxation of the

model. Constraints (14) ensure that each used batch is assigned to a machine. Constraints (15) and (16) determine the makespan.

3 Arc Flow approach

The arc flow approach has been used recently in classical optimization problems and allows modeling with a pseudo-polynomial number of variables and constraints. For a cutting-stock problem, [23] proposes a branch-and-price approach for an arc-flow formulation. Next, it was extended for the bin-packing problem in [24]. An alternative arc-flow formulation for the cutting-stock problem is proposed in [1] and [2], which uses a graph compression technique. These formulations were recently tested and compared in [9] against several other models and problem-specific algorithms on one-dimensional bin packing and cutting stock problems. The results show that the arc-flow formulation outperforms all other models. In [14] the arc-flow model and the one-cut model are compared for the one-dimensional cutting-stock problem, and reduction techniques for both approaches are presented.

For the scheduling area, we are only aware of two works that consider the arc-flow approach. In [12] the problem of scheduling a set of jobs on a set of identical parallel machines, with the aim of minimizing the total weighted completion time, $P||\sum W_jC_j$ is considered. In [17] the makespan minimization problem on identical parallel machines, $P||C_{max}$ is considered. It is important to note that these works do not consider more complex features in scheduling problems, such as batching machines, non-identical job sizes, and machine capacity.

The idea in this section is to formulate problem $P_m|s_j, B|C_{\max}$ as a problem of determining flows in graphs. With this goal, we initially define a directed graph $G = (V, A)$, in which each physical space of the batch with capacity B is represented by a node, i.e., $V = \{0, \ldots, B\}$. The set of directed arcs A is divided into three subsets: the set of *job arcs* A^J, the set of *loss arcs* A^L, and the set with a *feedback arc* A^F. Therefore, $A = A^J \cup A^L \cup A^F$. Each arc (i, j) of the subset A^J represents the existence of at least one job k of size s_k, such that $s_k = j - i$. The subset A^J is more specifically defined as $A^J := \{(i, j) : \exists k \in J, s_k = j - i \wedge i, j \in V \wedge i < j\}$. To compose valid paths and represent all possible solutions, it is necessary to include the *loss arcs* in G, which represent empty spaces at the end of a batch. The subset of arcs A^L is more specifically defined as $A^L := \{(i, B) : i \in V \wedge 0 < i < B\}$. Finally, the *feedback arc* is used to connect the last node to the first one, defined as $A^F := \{(B, 0)\}$.

The graph G is then replicated for each different processing time of the problem in our modeling approach. Each replicated graph will be referred to as an arc-flow structure for our problem. We consider $P := \{P_1, \ldots, P_\delta\}$ as the set with all the different processing times among all jobs, and $T := \{1, \ldots, \delta\}$ as the set of indexes corresponding to the arc-flow structures in the problem formulation.

A variable $w_{t,m}$ is created to determine the number of batches with processing time P_t that will be allocated on the machine m. Considering $NT_{\ell,t}$ ($NT^+_{\ell,t}$) as the number of jobs of size S_ℓ and processing time = P_t ($\leq P_t$), and NJ_t as the number of jobs with processing time P_t, our new formulation is presented below.

$f_{i,j,t}$: flow on *job arc* $(i, j) \in A^J$ in arc-flow structure t. The variable indicates the quantity of batches created with position i occupied by jobs with size $j - i$.

$y_{i,j,t}$: flow on the *loss arc* $(i, B) \in A^L$ in arc-flow structure t.

v_t : flow on the *feedback arc* in arc-flow structure t. The variable indicates the number of batches required with processing time P_t.

$z_{c,t}$: number of jobs with size c, not allocated in the batches with processing time smaller than or equal to P_t. Theses jobs are allowed to be allocated in the batches with processing time P_{t+1}.

$w_{t,m}$: number of batches with processing time P_t, allocated to machine m.

$$(\text{FLOW}_2) \quad \min\ C_{max} \tag{18}$$

$$\left(\sum_{(i,j)\in A^J} f_{i,j,t} + \sum_{(i,j)\in A^L} y_{i,j,t}\right) - \left(\sum_{(j,i)\in A^J} f_{j,i,t} + \sum_{(j,i)\in A^L} y_{j,i,t}\right) = \begin{cases} -v_t & \text{if } j = 0; \\ v_t & \text{if } j = B; \\ 0 & \text{if } 0 < j < B. \end{cases} \quad t \in T \tag{19}$$

$$NT_{c,t} - \sum_{\substack{(i,j)\in A^J: \\ j-i=c}} f_{i,j,t} = \begin{cases} z_{c,t} & \text{if } t = 1; \\ -z_{c,t-1} & \text{if } t = \delta; \\ z_{c,t} - z_{c,t-1} & \text{if } 1 < t < \delta. \end{cases} \quad c \in \{1..B\} \tag{20}$$

$$\sum_{m\in M} w_{t,m} \geq v_t \qquad t \in T \tag{21}$$

$$\sum_{t\in T} P_t w_{t,m} \leq C_{max} \qquad m \in M \tag{22}$$

$$f_{i,j,t} \leq min(NJ_t, NT^+_{j-i,t}),\ f_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^J \tag{23}$$

$$v_t \leq NJ_t,\ v_t \in \mathbb{Z} \qquad t \in T \tag{24}$$

$$y_{i,j,t} \leq NJ_t,\ y_{i,j,t} \in \mathbb{Z} \qquad t \in T, (i,j) \in A^L \tag{25}$$

$$z_{c,t} \leq NT^+_{c,t},\ z_{c,t} \in \mathbb{Z} \qquad t \in T : t < \delta, c \in \{1..B\} \tag{26}$$

$$w_{t,m} \in \mathbb{Z} \qquad t \in T, m \in M \tag{27}$$

The objective function (18) minimizes the makespan. The set of flow conservation constraints are defined by constraints (19). Constraints (20) ensure that all jobs are assigned and also control the number of jobs to be assigned to each arc-flow structure. Constraints (21) ensure that all batches used are assigned to a machine. Constraints (22) determine the makespan as the time required to finish processing the last batch on all machines. Constraints (23–27) define the domains of the variables and their respective upper bounds. We emphasize that (21) and (22) are the constraints that make it possible for the arc-flow model to handle batch allocation on parallel machines.

4 Computational results

The models presented in this chapter were compared through computational tests performed. The set was created by the authors of [4], who kindly sent them to us, to use in our work. Wee use the `CPLEX` version 12.7.1.0, configured to run in only one thread to not benefit from the processor parallelism. We used a computer with a 2.70GHz `Intel Quad-Core Xeon E5-2697 v2` processor and 64GB of RAM. The computational time to solve each instance was limited in 1800 seconds.

The set of test instances for problem $P_m|s_j, B|C_{\max}$ is the same considered in [4] for the $1|s_j, B|C_{\max}$ problem. For each job j, an integer processing time p_j and an integer job size s_j were generated from the respective uniform distribution depicted in Table 1. In total, 4200 instances were generated, 100 for each of the 42 different combinations of number and size of the jobs. We test each instance with three different numbers of parallel machines.

Table 1: Parameter settings.

Number of jobs (n_J)	Processing time (p_J)	Jobs size	Machine capacity (B)	Parallel machines (n_M)
10, 20, 50, 100 200, 300, 500	p_1: [1, 10] p_2: [1, 20]	s_1: [1, 10] s_2: [2, 4] s_3: [4, 8]	$B = 10$	2, 4, 8

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

Instance		(MILP)			(MILP$^+$)			(FLOW)		
Jobs	Type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
2 parallel machines										
10	p_1s_1	**18.76**	0.13	**0.00**	**18.76**	**0.01**	**0.00**	**18.76**	0.02	**0.00**
10	p_1s_2	**11.03**	0.05	**0.00**	**11.03**	**0.02**	**0.00**	**11.03**	**0.02**	**0.00**
10	p_1s_3	**22.13**	0.19	**0.00**	**22.13**	0.01	**0.00**	**22.13**	**0.00**	**0.00**
10	p_2s_1	**34.50**	0.12	**0.00**	**34.50**	**0.01**	**0.00**	**34.50**	0.03	**0.00**
10	p_2s_2	**21.71**	0.05	**0.00**	**21.71**	**0.02**	**0.00**	**21.71**	0.03	**0.00**
10	p_2s_3	**40.87**	0.17	**0.00**	**40.87**	**0.01**	**0.00**	**40.87**	**0.01**	**0.00**
20	p_1s_1	**34.27**	1308.41	5.54	**34.27**	**0.03**	**0.00**	**34.27**	0.04	**0.00**
20	p_1s_2	**18.83**	884.08	8.16	**18.83**	0.11	**0.00**	**18.83**	**0.04**	**0.00**
20	p_1s_3	**42.13**	1412.74	6.27	**42.13**	0.02	**0.00**	**42.13**	**0.01**	**0.00**
20	p_2s_1	**66.79**	1287.70	4.35	**66.79**	**0.03**	**0.00**	**66.79**	0.09	**0.00**
20	p_2s_2	**36.87**	651.70	7.05	**36.87**	0.15	**0.00**	**36.87**	**0.09**	**0.00**
20	p_2s_3	**79.82**	1395.83	5.60	**79.82**	0.02	**0.00**	**79.82**	**0.01**	**0.00**
50	p_1s_1	83.07	-	58.36	**82.30**	2.48	**0.00**	**82.30**	**0.08**	**0.00**
50	p_1s_2	46.56	-	59.68	**43.94**	529.33	0.52	**43.94**	**0.07**	**0.00**
50	p_1s_3	101.74	-	60.69	**101.30**	0.02	**0.00**	**101.30**	**0.01**	**0.00**
50	p_2s_1	159.08	-	61.30	**157.52**	5.12	**0.00**	**157.52**	**0.33**	**0.00**
50	p_2s_2	88.96	-	62.44	**84.32**	478.37	0.19	**84.32**	**0.55**	**0.00**
50	p_2s_3	192.95	-	64.02	**192.34**	0.03	**0.00**	**192.34**	**0.02**	**0.00**
100	p_1s_1	171.60	-	87.71	**159.78**	192.10	0.07	**159.78**	**0.11**	**0.00**
100	p_1s_2	98.19	-	86.49	**85.56**	1743.59	1.73	**85.56**	**0.10**	**0.00**
100	p_1s_3	206.66	-	86.52	**198.75**	0.15	**0.00**	**198.75**	**0.01**	**0.00**
100	p_2s_1	328.38	-	89.47	**305.58**	84.36	0.02	**305.58**	**0.42**	**0.00**
100	p_2s_2	188.69	-	88.60	163.39	1770.79	1.21	**163.31**	**1.58**	**0.00**
100	p_2s_3	398.94	-	89.28	**383.73**	0.20	**0.00**	**383.73**	**0.03**	**0.00**
200	p_1s_1	Unperformed			314.93	332.53	0.05	**314.92**	**0.07**	**0.00**
200	p_1s_2				167.44	-	1.58	**166.97**	**0.15**	**0.00**
200	p_1s_3				**393.36**	79.14	0.01	**393.36**	**0.02**	**0.00**
200	p_2s_1				599.00	495.03	0.05	**598.96**	**0.67**	**0.00**
200	p_2s_2				320.16	-	1.52	**318.85**	**3.43**	**0.00**
200	p_2s_3				**752.78**	42.39	**0.00**	**752.78**	**0.05**	**0.00**
300	p_1s_1				464.59	639.71	0.08	**464.54**	**0.10**	**0.00**
300	p_1s_2				250.62	-	1.89	**248.06**	**0.14**	**0.00**
300	p_1s_3				**587.49**	241.24	0.02	**587.49**	**0.02**	**0.00**
300	p_2s_1				897.09	764.46	0.05	**897.00**	**0.57**	**0.00**
300	p_2s_2				487.55	-	2.09	**481.61**	**3.25**	**0.00**
300	p_2s_3				**1123.96**	274.67	0.02	**1123.96**	**0.09**	**0.00**
500	p_1s_1				772.54	1084.33	0.11	**772.38**	**0.11**	**0.00**
500	p_1s_2				421.92	-	1.98	**415.76**	**0.17**	**0.00**
500	p_1s_3				**975.15**	382.95	0.02	**975.15**	**0.01**	**0.00**
500	p_2s_1				1483.02	1365.87	0.09	**1482.58**	**0.59**	**0.00**
500	p_2s_2				806.24	-	2.10	**794.00**	**2.78**	**0.00**
500	p_2s_3				**1851.16**	488.38	0.01	**1851.16**	**0.06**	**0.00**

We present in Table 2–4 comparison results among the arc flow formulation proposed in this work and another two from the literature. All values presented are the average results computed over the instances of the same configuration, as described in Table 1.

The comparative tests clearly show that formulation (FLOW) is superior to (MILP) and (MILP$^+$), especially when the number of jobs increases. Model (FLOW) did not prove the optimality of only one instance from the set of test problems. For instances with 20 jobs or less, (MILP$^+$) can solve some instances in less computational time than (FLOW), but the difference between times is always a fraction of a second. Additionally, the duality gaps shown for (MILP) reveal the difficulty in obtaining good lower bounds.

Unlike what we have with models (MILP) and (MILP$^+$), the number of variables in (FLOW) does not grow when the number of jobs increases. Moreover, the flow graph does not change in this

Table 3: Computational results for $P_m|s_j.B|C_{\max}$ - 4 parallel machines.

Instance		(MILP)			(MILP$^+$)			(FLOW)		
jobs	type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
4 parallel machines										
10	p_1s_1	**10.87**	0.16	**0.00**	**10.87**	**0.02**	**0.00**	**10.87**	**0.02**	**0.00**
10	p_1s_2	**9.49**	0.10	**0.00**	**9.49**	**0.01**	**0.00**	**9.49**	**0.01**	**0.00**
10	p_1s_3	**12.18**	0.25	**0.00**	**12.18**	0.02	**0.00**	**12.18**	**0.01**	**0.00**
10	p_2s_1	**20.26**	0.16	**0.00**	**20.26**	**0.02**	**0.00**	**20.26**	0.03	**0.00**
10	p_2s_2	**18.68**	0.11	**0.00**	**18.68**	**0.01**	**0.00**	**18.68**	0.02	**0.00**
10	p_2s_3	**22.67**	0.23	**0.00**	**22.67**	0.02	**0.00**	**22.67**	**0.01**	**0.00**
20	p_1s_1	**17.47**	1316.19	8.11	**17.47**	**0.05**	**0.00**	**17.47**	0.06	**0.00**
20	p_1s_2	**10.43**	56.14	0.49	**10.43**	0.32	**0.00**	**10.43**	**0.05**	**0.00**
20	p_1s_3	**21.29**	1629.93	11.78	**21.29**	0.03	**0.00**	**21.29**	**0.01**	**0.00**
20	p_2s_1	**33.95**	1122.49	5.29	**33.95**	**0.07**	**0.00**	**33.95**	0.14	**0.00**
20	p_2s_2	**20.51**	92.62	0.64	**20.51**	0.35	**0.00**	**20.51**	**0.14**	**0.00**
20	p_2s_3	**40.21**	1731.24	12.27	**40.21**	0.05	**0.00**	**40.21**	**0.02**	**0.00**
50	p_1s_1	42.69	-	70.03	**41.43**	2.54	**0.00**	**41.43**	**0.11**	**0.00**
50	p_1s_2	23.76	-	58.83	**22.18**	269.31	0.56	**22.18**	**0.26**	**0.00**
50	p_1s_3	51.85	-	71.91	**50.90**	0.05	**0.00**	**50.90**	**0.01**	**0.00**
50	p_2s_1	81.23	-	71.19	**78.97**	0.90	**0.00**	**78.97**	**0.80**	**0.00**
50	p_2s_2	45.55	-	57.37	**42.38**	283.70	0.19	**42.38**	**1.41**	**0.00**
50	p_2s_3	97.89	-	73.39	**96.40**	0.07	**0.00**	**96.40**	**0.04**	**0.00**
100	p_1s_1	93.06	-	93.44	**80.09**	82.33	0.05	**80.09**	**0.18**	**0.00**
100	p_1s_2	50.26	-	81.80	43.06	1409.33	1.67	**43.04**	**0.26**	**0.00**
100	p_1s_3	110.60	-	92.94	**99.64**	0.55	**0.00**	**99.64**	**0.02**	**0.00**
100	p_2s_1	177.17	-	93.54	**153.03**	51.98	0.02	**153.03**	**1.83**	**0.00**
100	p_2s_2	96.18	-	86.63	82.03	1679.11	1.32	**81.88**	**3.12**	**0.00**
100	p_2s_3	213.47	-	93.38	**192.11**	0.52	**0.00**	**192.11**	**0.05**	**0.00**
200	p_1s_1	Unperformed			157.71	209.19	0.06	**157.70**	**0.16**	**0.00**
200	p_1s_2				84.05	1788.54	1.69	**83.67**	**0.53**	**0.00**
200	p_1s_3				**196.93**	38.03	0.01	**196.93**	**0.02**	**0.00**
200	p_2s_1				299.78	396.85	0.06	**299.75**	**21.23**	**0.00**
200	p_2s_2				160.95	-	1.94	**159.68**	**31.02**	**0.01**
200	p_2s_3				**376.64**	59.36	0.01	**376.64**	**0.07**	**0.00**
300	p_1s_1				232.56	422.79	0.09	**232.52**	**0.17**	**0.00**
300	p_1s_2				126.22	-	2.40	**124.28**	**0.33**	**0.00**
300	p_1s_3				**293.99**	146.32	0.03	**293.99**	**0.02**	**0.00**
300	p_2s_1				448.84	568.62	0.06	**448.79**	**1.19**	**0.00**
300	p_2s_2				244.96	-	2.49	**241.07**	**21.59**	**0.00**
300	p_2s_3				**562.24**	230.89	0.02	**562.24**	**0.11**	**0.00**
500	p_1s_1				386.62	1009.64	0.15	**386.47**	**0.17**	**0.00**
500	p_1s_2				211.91	-	2.31	**208.12**	**0.40**	**0.00**
500	p_1s_3				**487.84**	266.44	0.03	**487.84**	**0.02**	**0.00**
500	p_2s_1				741.93	1306.98	0.12	**741.56**	**1.91**	**0.00**
500	p_2s_2				405.30	-	2.58	**397.30**	**134.68**	**0.01**
500	p_2s_3				**925.84**	345.92	0.02	**925.84**	**0.07**	**0.00**

Table 4: Computational results for $P_m|s_j.B|C_{\max}$ - 8 parallel machines.

Instance		(MILP)			(MILP$^+$)			(FLOW)		
jobs	type	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap	$C_{\max}$	$T(s)$	Gap
8 parallel machines										
10	p_1s_1	**9.54**	0.23	**0.00**	**9.54**	**0.01**	**0.00**	**9.54**	0.02	**0.00**
10	p_1s_2	**9.49**	0.25	**0.00**	**9.49**	**0.01**	**0.00**	**9.49**	0.02	**0.00**
10	p_1s_3	**9.42**	0.33	**0.00**	**9.42**	**0.01**	**0.00**	**9.42**	**0.01**	**0.00**
10	p_2s_1	**18.55**	0.21	**0.00**	**18.55**	**0.01**	**0.00**	**18.55**	0.02	**0.00**
10	p_2s_2	**18.68**	0.24	**0.00**	**18.68**	**0.01**	**0.00**	**18.68**	0.02	**0.00**
10	p_2s_3	**18.27**	0.34	**0.00**	**18.27**	**0.01**	**0.00**	**18.27**	**0.01**	**0.00**
20	p_1s_1	**10.51**	276.62	2.44	**10.51**	0.09	**0.00**	**10.51**	**0.06**	**0.00**
20	p_1s_2	**9.81**	2.76	**0.00**	**9.81**	0.07	**0.00**	**9.81**	**0.03**	**0.00**
20	p_1s_3	11.61	760.27	7.24	**11.60**	0.15	**0.00**	**11.60**	**0.01**	**0.00**
20	p_2s_1	**20.76**	328.01	3.34	**20.76**	**0.13**	**0.00**	**20.76**	0.18	**0.00**
20	p_2s_2	**19.52**	2.80	**0.00**	**19.52**	0.08	**0.00**	**19.52**	**0.04**	**0.00**
20	p_2s_3	22.31	958.29	8.28	**22.30**	0.26	**0.00**	**22.30**	**0.04**	**0.00**
50	p_1s_1	22.30	-	55.90	**20.96**	2.99	**0.00**	**20.96**	**0.25**	**0.00**
50	p_1s_2	12.83	1783.20	27.02	**11.77**	850.12	4.12	**11.77**	**0.39**	**0.00**
50	p_1s_3	26.78	-	62.67	**25.71**	0.10	**0.00**	**25.71**	**0.02**	**0.00**
50	p_2s_1	42.41	-	55.68	**39.72**	**1.25**	**0.00**	**39.72**	2.47	**0.00**
50	p_2s_2	24.41	1775.39	28.90	22.46	1198.77	3.91	**22.45**	**11.60**	**0.00**
50	p_2s_3	50.33	-	61.21	**48.45**	0.17	**0.00**	**48.45**	**0.08**	**0.00**
100	p_1s_1	59.57	-	96.84	**40.34**	51.78	0.05	**40.34**	**0.19**	**0.00**
100	p_1s_2	28.80	-	84.72	21.82	872.63	2.04	**21.75**	**1.49**	**0.00**
100	p_1s_3	69.72	-	98.03	**50.07**	0.22	**0.00**	**50.07**	**0.03**	**0.00**
100	p_2s_1	123.38	-	97.70	**76.81**	59.45	0.01	**76.81**	**10.88**	**0.00**
100	p_2s_2	57.82	-	94.95	41.34	1251.21	1.45	**41.23**	**37.99**	**0.03**
100	p_2s_3	139.99	-	98.19	**96.34**	0.81	**0.00**	**96.34**	**0.11**	**0.00**
200	p_1s_1	Unperformed			79.13	213.02	0.12	**79.10**	**0.15**	**0.00**
200	p_1s_2				42.41	1599.90	1.97	**42.10**	**0.95**	**0.00**
200	p_1s_3				**98.74**	2.72	**0.00**	**98.74**	**0.02**	**0.00**
200	p_2s_1				150.18	341.94	0.08	**150.14**	**54.67**	**0.01**
200	p_2s_2				81.27	1796.74	2.64	**80.08**	**141.71**	**0.04**
200	p_2s_3				**188.53**	21.83	0.01	**188.53**	**0.14**	**0.00**
300	p_1s_1				116.58	480.57	0.16	**116.51**	**18.20**	**0.01**
300	p_1s_2				63.40	1779.89	2.44	**62.39**	**0.57**	**0.00**
300	p_1s_3				**147.25**	94.40	0.03	**147.25**	**0.02**	**0.00**
300	p_2s_1				224.76	641.81	0.12	**224.61**	**2.84**	**0.00**
300	p_2s_2				123.61	-	3.20	**120.78**	**358.94**	**0.13**
300	p_2s_3				**281.31**	115.95	0.02	**281.31**	**0.31**	**0.00**
500	p_1s_1				193.74	1067.41	0.29	**193.53**	**0.24**	**0.00**
500	p_1s_2				106.63	1787.69	2.72	**104.38**	**1.67**	**0.00**
500	p_1s_3				**244.11**	134.96	0.03	**244.11**	**0.04**	**0.00**
500	p_2s_1				371.50	1136.26	0.20	**371.03**	**2.08**	**0.00**
500	p_2s_2				203.42	-	2.83	**198.96**	**376.70**	**0.09**
500	p_2s_3				**463.16**	189.74	0.02	**463.16**	**0.24**	**0.00**

case. Only the bounds on the variables change. The flow graphs of two distinct instances will be the same if the settings in the parameters Processing Time, Job Size and Machine Capacity are the same. In fact, this is a very important characteristic of the flow approach. We finally note that the computational time to construct the graphs for the flow formulation was not considered in these times. However, the maximum time to construct a graph for any instance in our experiments was 0.008 second.

The results show that instances of configuration s_2 require more computational time and are more difficult compared to the other instances for all formulations. The reason for this is the small sizes of the jobs when compared to the machine capacity, which allows more combinations of assignment to a batch.

Conclusion

In this paper we propose a new arc-flow formulation for minimizing makespan on parallel batch machines, considering non-identical job sizes. The computational results reveal that this new approach is much more efficient than those previously published in the literature. It is able to solve instances up to 500 jobs, which have never been solved before, with low computational times. Even for the most difficult instances, for which the model failed to prove optimality, the results are very close to the optimal with gaps between 0.13-0.01%. One of the best advantages of the arc-flow model is that the number of variables does not increase if the number of jobs of the instance increases.

As future work, it is interesting to investigate whether this approach can be applied to other variants of scheduling problems, such as considering incompatible families or jobs with non-identical release times.

References

[1] F. Brandão and J. P. Pedroso. Bin packing and related problems: General arc-flow formulation with graph compression. *Computers and Operations Research*, 69:56–67, may 2016.

[2] F. D. A. Brandão. *Cutting & Packing Problems: General Arc-flow Formulation with Graph Compression*. PhD thesis, Universidade do Porto, 2017.

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

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

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

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

[7] S. Chung, Y. Tai, and W. Pearn. 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, sep 2009.

[8] P. Damodaran, N. S. Hirani, and M. C. V. Gallego. Scheduling identical parallel batch processing machines to minimise makespan using genetic algorithms. *European Journal of Industrial Engineering*, 3(2):187, sept 2009.

[9] M. Delorme, M. Iori, and S. Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, nov 2016.

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

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

[12] A. Kramer, M. Dell'Amico, and M. Iori. Enhanced arc-flow formulations to minimize weighted completion time on identical parallel machines. *European Journal of Operational Research*, 275(1):67–79, may 2019.

[13] C.-Y. Lee, R. Uzsoy, and L. A. Martin-Vega. Efficient Algorithms for Scheduling Semiconductor Burn-In Operations. *Operations Research*, 40(4):pp. 764–775, aug 1992.

[14] J. Martinovic, G. Scheithauer, and J. M. Valério de Carvalho. A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, 266(2):458–471, apr 2018.

[15] M. Mathirajan and A. Sivakumar. A 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, jan 2006.

[16] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6):583–599, jan 2011.

[17] M. Mrad and N. Souayah. An Arc-Flow Model for the Makespan Minimization Problem on Identical Parallel Machines. *IEEE Access*, 6:5300–5307, jan 2018.

[18] C. N. Potts and M. Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249, jan 2000.

[19] Y. Tai. *The Study on the Production Scheduling Problems for Liquid Crystal Display Module Assembly factories*. PhD thesis, National Chiao Tung University, jul 2008.

[20] R. S. Trindade. *Modelling Batch Processing Machines Problems with Symmetry Breaking and Arc Flow Formulation*. PhD thesis, Universidade Federal do Rio de Janeiro, 2019.

[21] R. S. Trindade, O. C. B. de Araújo, M. H. C. Fampa, and F. M. Müller. Modelling and symmetry breaking in scheduling problems on batch processing machines. *International Journal of Production Research*, 56(22):7031–7048, nov 2018.

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

[23] J. Valério de Carvalho. Exact solution of cutting stock problems using column generation and branch-and-bound. *International Transactions in Operational Research*, 5(1):35–44, 1998.

[24] J. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, jan 1999.