São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

Arc Flow Model for Scheduling Jobs on Identical Parallel Machines to Minimize the Makespan

Alisson Michel Sganzerla, Olinto César Bassi de Araújo
Departamento de Engenharia de Produção, Universidade Federal de Santa Maria
Prédio 8 - INPE - Sala 2057, Av. Roraima, 1000, Santa Maria - RS
alisson.sganzerla@acad.ufsm.br, olinto@ctism.ufsm.br

Renan Spencer Trindade
LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris
Bâtiment Alan Turing, 1 rue Honoré d'Estienne d'Orves, 91128, Palaiseau, France
rst@lix.polytechnique.fr

RESUMO

Este trabalho apresenta um modelo de fluxo em arcos para o problema de programação de tarefas em máquinas paralelas com o objetivo de minimizar o tempo total para processar todas as tarefas na máquina mais carregada (*makespan*). No modelo proposto, a função objetivo minimiza o maior valor relativo a um vértice no qual o fluxo inicial é injetado na rede. Para tornar essa abordagem coerente, a origem do fluxo ocorre nos vértices associados ao intervalo entre os limitantes inferior e superior. Essa abordagem implica na inversão da direção do fluxo, tornando o vértice zero o destino final, diferente da abordagem convencional em que o vértice zero é a origem do fluxo. O modelo foi testado utilizando CPLEX v20.1 para resolver 3500 instâncias considerando diferentes proporções entre o número de tarefas e máquinas. Os resultados computacionais demonstraram que foi possível obter soluções ótimas em um tempo com ganho percentual médio de 400% quando comparado com o modelo da literatura.

PALAVRAS CHAVE. Programação Inteira, Modelo em Fluxos de Arcos, Programação de Tarefas.

ABSTRACT

This paper presents an arc flow model for the job scheduling problem on parallel machines to minimize the total time to process all jobs on the most loaded machine (makespan). In the proposed model, the objective function minimizes the largest value associated with a vertex where the initial flow is injected into the network. To make this approach consistent, the origin of the flow occurs at nodes related to the interval between the lower and upper bounds. This approach implies reversing the flow direction, making node zero the final destination, unlike the conventional approach where node zero is the origin of flow. We tested the model using CPLEX v20.1 to solve 3500 instances, considering different ratios between the number of jobs and machines. The computational results showed that obtaining optimal solutions with an average percentage time gain of 400% compared to the literature model was possible.

KEYWORDS. Integer Programming, Arc-Flow Model, Job Scheduling Problem.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

1. Introduction

In recent decades, arc flow-based models have achieved significant attention for their performance in tackling various classical optimization problems. This approach originated from the pioneering work of Ford and Fulkerson [1958], presenting a pseudo-polynomial model for a logistics problem utilizing arc flow formulations. Shapiro [1968] later showed that a network structure could effectively represent the dual version of the knapsack problem. Wolsey [1977] proposed network models for packing and covering problems, exploring their dual versions. However, this formulation concept has gained popularity only in recent years, particularly following the work of Valério de Carvalho [1999], which presented competitive computational results using arc flow formulations for the bin packing problem.

Therefore, arc flow-based formulations have emerged as a practical and effective optimization approach for addressing a wide range of applications, such as the packing problem (a) [Brandão and Pedroso, 2016], guillotine cutting ((b) [Macedo et al., 2010], (c) [Mrad, 2015], (d) [Martinovic et al., 2018], (e) [Martinovic et al., 2020], (f) [da Silva et al., 2023]), production logistics ((g) [Ramos et al., 2022]) and scale of rail service (h) [Hoffmann and Buscher, 2019], among others. A particular interest in using arc flow-based models for job scheduling problems has also arisen in recent years, including ((i) [Mrad and Souayah, 2018], (j) [Kramer et al., 2019], (k) [Trindade et al., 2020], (l) [Silva et al., 2021], (m) [Gharbi and Bamatraf, 2022]). These studies are presented in chronological order in Figure 1.



Figure 1: Timeline of publications with arc flow modeling.

This paper presents a new arc flow model formulation to address the classical problem of scheduling jobs on identical parallel machines to minimize the time required to process all jobs on the machine with the highest makespan. This problem is denoted as $P||Cmax$ in the three-field classification introduced by Graham et al. [1979]. Several approaches have been used to solve this problem to optimality, such as an exact algorithm based on cutting plans [Mokotoff, 2004], an Integer Linear Program approach with column generation and branch and price [Dell'Amico et al., 2008]. Dell'Amico et al. [2008] also presents a heuristic that uses Scatter Search. Other heuristics are suggested as Simulated Annealing [Laha, 2012], and Cuckoo Search [Laha and Gupta, 2018]. An arc flow formulation was first proposed for this problem in Mrad and Souayah [2018], showing promising results compared to other approaches. The wide array of methods and techniques to address the $P||Cmax$ problem underscores the current demand for efficient, high-quality solutions.

In our work, we propose a different arc flow formulation for this problem, with a new methodology for generating arcs, associating the objective function with the vertex index where the initial flow is injected into the network. Then, the origin of the flow is selected within a range of candidate vertices defined by a lower and an upper bound. This approach contrasts with the procedures adopted in Mrad and Souayah [2018], where the flow source is typically designated at the initial node within the network.

The remainder of this paper is organized as described as follows. Section 2 presents a detailed description of the $P||Cmax$ problem and the reference model in the literature developed with the arc flow technique. Section 3 presents the model proposed in this paper, highlighting the

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

differences with the reference model. Section 4 presents and discusses the computational results regarding the model comparison. Finally, at the end, the contributions of this work are discussed, as well as suggestions for future work.

2. Job Scheduling Problems

Since the 1960s, researchers have extensively investigated job scheduling in parallel machine problems to develop novel solutions that improve efficiency and productivity across diverse industrial sectors. This class of problems presents distinct variations, which depend on the limitations of available resources or the optimization criteria considered.

Our work considers $P||Cmax$, defined as follows. Given a set of n jobs in $J = \{1, ..., n\}$, and m identical parallel machines in $M = \{1, ..., m\}$ such that $n > m \geq 2$. Each job $i \in J$ has a processing time p_i, and each machine can process only one job simultaneously without interruptions. The goal is to minimize the time needed to complete all jobs, also known as makespan. This problem is classified as NP-hard [Garey and Johnson, 1978], and the complexity and relevance of this problem stimulate the search for efficient algorithms and heuristics.

2.1. Model proposed in Mrad and Souayah [2018]

Mrad and Souayah [2018] proposed an arc flow-based formulation for the $P||Cmax$ problem as a Mixed Integer Linear Programming (MILP). The purpose is to find disjoint paths using an associated acyclic direct graph representing all valid scheduling options. This model establishes a relationship between the dual version of the bin packing problem and the $P||Cmax$ problem. In the present study, this model is referred to as $M\&S$.

It is worth noting that Gharbi and Bamatraf [2022] present a model equivalent to $M\&S$. The computational results reported by Gharbi and Bamatraf are mainly due to the methodology to calculate the value of the upper bound rather than structural changes in the model.

Before constructing the graph, an upper bound UB is obtained through a greedy heuristic Longest Processing Time [Graham, 1969]. This heuristic consists of sorting the jobs in non-increasing order based on their processing times and then distributing each job to the machines with the lowest makespan at the time. A lower bound LB is also obtained by calculating the ratio of the sum of the processing times p_i of all jobs $i \in J$, and the quantity m of available machines. This formula is expressed as: $LB = \lceil \frac{\sum p_i}{m} \rceil$.

The associated graph to the model has the following structure: $G = (V, A' \cup A'')$. The set V consists of vertices represented by $\{0, ..., UB\}$. The set A' consists of binary arcs (i, j), representing each job k with processing time $p_k = j - i$. The set A'' consists of integer virtual arcs connecting all vertices $i \in V \setminus \{0\}$ to the vertex UB, ensuring the existence of the path between any vertex.

Symmetry-breaking rules are applied to reduce the unnecessary number of generated arcs and vertices. This reduction in the size of the graph contributes to a decrease in computational time and in handling instances with more jobs. These rules require the jobs to be in non-increasing order based on processing time. Then, an arc can only be generated from vertex zero or from a vertex already connected to an existing arc.

The graph generation process is described in Algorithm 1. To exemplify, a set of jobs $J = \{1, 2, 3, 4\}$ with processing times $P = \{5, 3, 3, 2\}$ and bounds $UB = 8$ and $LB = 6$ is considered and illustrated in Figure 2. The algorithm is started with two predefined vertices, $\{0, UB\}$. For each job $i \in J$, the arcs are generated iteratively, not exceeding the UB boundary and considering only the vertices generated in the previous iterations. When generating a new arc $(k, k + p_i)$, the vertex

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

Algorithm 1
Graph generation process $M\&S$

$V \leftarrow \{0, UB\}$
$A' \leftarrow \emptyset$
for $i \in J$ **do**
  $NewV \leftarrow \emptyset$
  for $k \in V \setminus \{UB\}$ **do**
    if $k + p_i \leq UB$ **then**
      $A' \leftarrow A' \cup (i, k, k + p_i)$
      $NewV \leftarrow NewV \cup (k + p_i)$
    end if
  end for
  $V \leftarrow V \cup NewV$
end for
$A'' \leftarrow \emptyset$
for $k \in V \setminus \{0, UB\}$ **do**
  $A'' \leftarrow A'' \cup (k, UB)$
end for

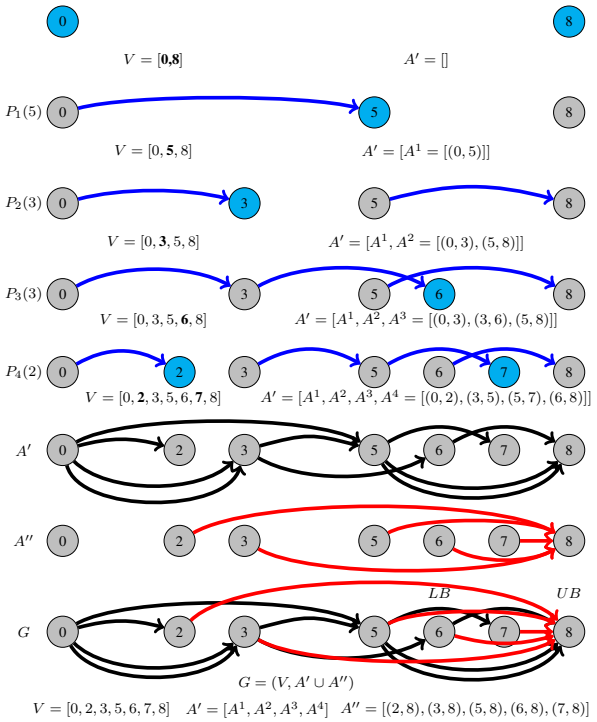


Figure 2: Example of the graph generation process for $M\&S$.

p_i is created only if it does not already exist. Arcs A' are indicated in blue in the respective figure. Then, the arcs of the set A'' are generated, linking all intermediate vertices up to UB, indicated in the figure by red lines. Thus, the complete graph with both sets A' and A'' is obtained.

$$(M\&S): \textbf{Minimize } z \tag{1}$$

S.t.:

$$z \geq j \cdot x_{kij} \qquad \forall (k,i,j) \in A', j \geq LB \tag{2}$$

$$\sum_{(k,0,j)\in A'} x_{k0j} = m \tag{3}$$

$$\sum_{(k,i,j)\in A'} x_{kij} - \sum_{(k,j,i)\in A'} x_{kji} - y_{j,UB} = 0 \qquad \forall j \in V \setminus \{0, UB\} \tag{4}$$

$$\sum_{(k,i,j)\in A'} x_{kij} = 1 \qquad \forall k \in J \tag{5}$$

$$x_{kij} \in \{0,1\} \qquad \forall (k,i,j) \in A' \tag{6}$$

$$y_{i,UB} \in \mathbb{N} \qquad \forall (i, UB) \in A'' \tag{7}$$

$$LB \leq z \leq UB \tag{8}$$

In the $M\&S$ formulation, the variables x_{kij} assume value 1 when the arc (i, j) of job k is used in the solution; and 0 otherwise, $\forall (k, i, j) \in A'$. The integer variable $y_{i,UB}$ represents the

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

number of machines that have processing time equal to i. The variable z represents the makespan. The formulation is presented below:

The objective function (1) minimizes the makespan, represented by variable z. The constraint (2) corresponds to the min/max strategy, where the variable z represents the time at which the processing of the last job is completed. The constraint (3) determines that the amount of flow injected in vertex 0 is equal to the amount of machines. The constraint (4) guarantees flow conservation for all intermediate vertices of the set V. The constraint (5) ensures that all jobs k must be processed. The constraints (6), (7) and (8) sets the domain of variables. The final solution is represented in the graph by m disjoint paths between vertex 0 and UB, where each job appears on a single path.

3. Proposed mathematical model

The proposed approach in this paper eliminates the necessity for a specific variable to represent the time of the machine with the highest time and the corresponding constraint, which restricts its value to a network vertex. Instead, the objective function value is ascertained based on the vertex with the minimum associated value, where the initial flow is injected into the network. The source of the flow is selected from the potential vertices within the range defined by the lower and upper bounds. This modification implies a reversal of the flow direction in the graph, thereby designating the vertex with an index of zero as the final destination. This approach differs from the conventional approach, in which the first vertex of the network is defined as the origin of the flow.

3.1. Lower and upper bounds

Although there are more sophisticated approaches available to obtain bounds for the problem $P||Cmax$, we chose to keep the same approach used by Mrad and Souayah [2018] as described in Section 2 to allow a fair comparison between the models. However, implementing stricter bounds can yield significant advantages, including a decrease in the number of arcs generated and consequent enhancement of model performance overall. We are aware that it is possible to use alternative methodologies, such as the heuristic Variable Neighborhood Search proposed by Gharbi and Bamatraf [2022] for the upper bounds, or the set of rules for lower bounds described by Dell'Amico and Martello [1995].

3.2. New formulation

The proposed formulation called STA, corresponds to a Mixed Integer Linear Programming model that uses flows in arcs to treat the $P||Cmax$ problem. This formulation represents an improvement over the $M\&S$ model, as suggested by the results presented in Section 4.

The construction of the associated graph to the STA model follows the same symmetry-breaking rules described in Section 2. The main difference is that all arcs (i, j) of the set A' are now created in the opposite direction, i.e. connecting the vertices (j, i). Thus, the arcs of the set A'' are also created differently, connecting each node to its immediate predecessor. The Algorithm 2 describes the new generation process.

To illustrate the process of graph creation, we employ the same instance as discussed in the preceding section, with jobs $J = \{1, 2, 3, 4\}$ and the respective processing times $P = \{5, 3, 3, 2\}$. As established by the first symmetry-breaking rule, the jobs are already sorted in non-increasing order according to processing time. The second rule, which describes the construction of the arcs A' and the vertices V, is detailed in Algorithm 2 and can be visualized in Figure 3. After this process, the set V is sorted, and the arcs of the set A'' are created by connecting each vertex $i \in V$ with its respective successor.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

Algorithm 2
Graph generation process STA

1: $V \leftarrow \{0, UB\}$
2: $A' \leftarrow \emptyset$
3: **for** $i \in J$ **do**
4: $\quad NewV \leftarrow \emptyset$
5: $\quad$ **for** $k \in V \setminus \{UB\}$ **do**
6: $\quad\quad$ **if** $k + p_i \leq UB$ **then**
7: $\quad\quad\quad A' \leftarrow A' \cup (k + p_i, k)$
8: $\quad\quad\quad NewV \leftarrow NewV \cup (k + p_i)$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad V \leftarrow V \cup NewV$
12: **end for**
13: $A'' \leftarrow \emptyset$
14: $lastNode \leftarrow 0$
15: $V \leftarrow Sort(V)$
16: **for** $k \in V \setminus \{0\}$ **do**
17: $\quad A'' \leftarrow A'' \cup (k, lastNode)$
18: $\quad lastNode \leftarrow k$
19: **end for**

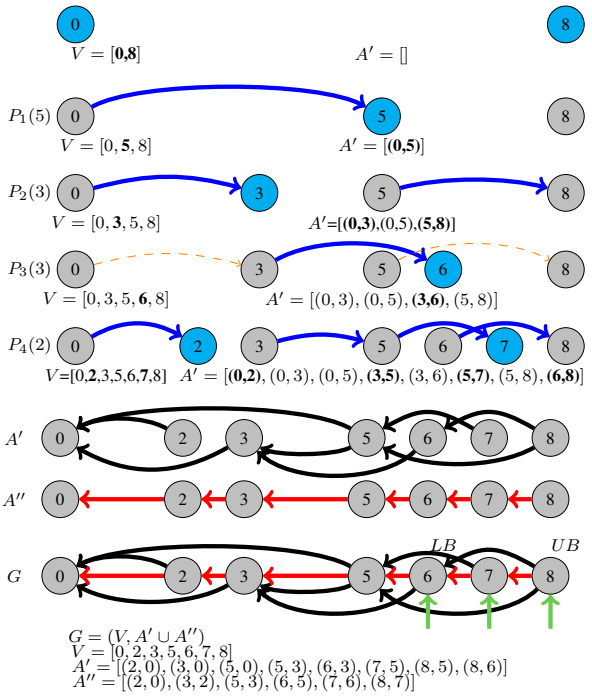


Figure 3: Example of the graph generation process for STA.

One notable distinction between the mathematical model STA and the model $M\&S$ is the use of integer variables to denote the arcs A'. This approach excludes repeated arcs $(0-3, 5-8)$ from the graph, which can be identified in orange in our example.

The image also illustrates the graphical representation of the A'' set, with the corresponding arcs highlighted in red. Moreover, the final graph illustrates the candidate vertices to set the makespan between the lower and upper bound, indicated by green arrows.

The processing times p_j are reclassified and grouped into a novel set denoted as $\mathcal{P} = \bigcup_{j \in J} p_j$, encompassing the union of all individual p_j values. Additionally, a new set $D = \{k : k \in 0..|\mathcal{P}|\}$ represents the indexes associated with the grouped processing times. Each element $k \in D$ is related to the quantity q_k of jobs that share identical processing times denoted as p'_k.

The mathematical model STA utilizes three sets of variables: the integer variable x_{ij} indicates the utilization of one or more jobs with a processing time equal to $j - i$. Meanwhile, the integer variable y_{ij} represents the virtual arcs originating from A'' and guarantees the existence of a path between any node in the graph. Furthermore, the binary variable w_j denotes the vertex where the initial flow is injected into the network.

The objective function denoted by (9) aims to minimize the largest vertex j linked to w_j, representing the makespan value. Consequently, the objective function minimizes the index of the vertex at which the flow initiates. The constraint (10) ensures that only one candidate vertex w_j is set as the source of the flow in the network. The constraint (11) guarantees flow conservation. The constraint (12) determines that all jobs are processed. The constraints (13), (14) and (15) correspond to the variable domain constraints. The final solution comprises a set of m disjoint paths in the graph, but this time connecting the vertex w to the vertex 0. The formulation is presented below:

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

$$(STA): \textbf{Minimize} \sum_{j \in V, j \geq LB} j \cdot w_j \tag{9}$$

S.t.:

$$\sum_{j \in V, j \geq LB} w_j = 1 \tag{10}$$

$$\sum_{(j,i) \in A'} x_{ji} - \sum_{(i,j) \in A'} x_{ij} + \sum_{(j,i) \in A''} y_{ji} - \sum_{(i,j) \in A''} y_{ij} + \begin{cases} 0 & \text{if} \quad j < LB \\ m \cdot w_j & \text{if} \quad j \geq LB \end{cases} = 0 \qquad \forall j \in V \setminus \{0\} \tag{11}$$

$$\sum_{(j,i) \in A', j - i = p'_k} x_{ji} = q_k \qquad \forall k \in D \tag{12}$$

$$x_{ji} \in \mathbb{N} \qquad \forall (i,j) \in A' \tag{13}$$

$$y_{ji} \in \mathbb{N} \qquad \forall (i,j) \in A'' \tag{14}$$

$$w_j \in \{0,1\} \qquad \forall j \in V, j \geq LB \tag{15}$$

4. Computational results

This section presents the computational results of tests performed with the $M\&S$ and STA models. We evaluate the performance of the models, both implemented in Python, using the CPLEX v20.1 with the default configuration. The experiment was performed on a computer equipped with an Intel Quad-Core Xeon E5-2697 3.5 GHz processor and 64 GB of RAM, running the Ubuntu 18.04 operating system.

4.1. Instances

To perform the tests, 3500 instances were generated following the characteristics described by Mrad and Souayah [2018] and they are available at the public link https://github.com/sganzerla/instances. These instances were divided into 5 groups $\{2.0, 2.25, 2.5, 2.75, 3.0\}$, taking into account the ratio between the amount of jobs n and machines m. Each group consists of seven C classes, and the processing times were generated according to the following distributions:

- $C1$: uniform discrete distribution in the interval $\{1, 100\}$
- $C2$: uniform discrete distribution in the interval $\{20, 100\}$
- $C3$: uniform discrete distribution in the interval $\{50, 100\}$
- $C4$: normal distribution with mean $\mu = 100$ and standard deviation $\sigma = 20$
- $C5$: normal distribution with mean $\mu = 100$ and standard deviation $\sigma = 50$
- $C6$: uniform discrete distribution in the interval $\{n, 4n\}$
- $C7$: normal distribution with mean $\mu = 4n$ and standard deviation $\sigma = n$

In each class, 10 (n/m) pairs are included according to:

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

- $(n/m) \in \{2.0, 2.5\}$, with n varying in $\{20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$
- $(n/m) \in \{2.25, 3.0\}$, with n varying in $\{36, 54, 72, 90, 108, 126, 144, 162, 180, 198\}$
- $(n/m) \in \{2.75\}$, with n varying in $\{22, 44, 66, 88, 110, 132, 154, 176, 198, 220\}$

4.2. Results

We summarized and grouped the results of the 3500 tested instances into five groups (n/m). Each group consisted of 700 instances, based on the ratio between the number of jobs and machines. Table 1 presents the average, maximum, and total time to solve the 700 instances in each group for which an optimal solution could be found. In addition, the table includes the average percentage time gain between the STA model and the $M\&S$ model, represented as $(\Delta\%)$. This difference was calculated using equation: $\Delta\% = \frac{(STA - M\&S)}{\min(STA, M\&S)} \times 100$.

n/m	$M\&S$				STA						
	#opt	*seconds*			*#opt*	*seconds*					
		avg	*max*	*total*		*avg*	$\Delta\%$	*max*	$\Delta\%$	*total*	$\Delta\%$
2.00	700	**0.38**	10.98	**267.10**	700	0.60	57.89	**8.36**	**-31.34**	422.85	58.31
2.25	700	16.36	875.79	11449.24	700	**2.72**	**-501.47**	**120.72**	**-625.47**	**1906.30**	**-500.60**
2.50	699	28.05	979.84	19610.18	**700**	**4.49**	**-524.72**	**137.39**	**-613.18**	**3138.11**	**-524.90**
2.75	697	52.23	1752.07	36407.79	**700**	**7.07**	**-638.76**	**277.49**	**-531.40**	**4946.73**	**-636.00**
3.00	700	10.21	575.13	7143.73	700	**5.81**	**-75.73**	**277.76**	**-107.06**	**4064.20**	**-75.77**
avg	-	21.45	838.76	14975.61	-	**4.14**	**-418.12**	**164.35**	**-410.35**	**2895.64**	**-417.18**
total	3496	-	-	-	**3500**	-	-	-	-	-	-

Table 1: Computational results for the $M\&S$ and STA models.

Firstly, it is worth noting that all instances were solved optimally using the STA method. However, the $M\&S$ method failed to obtain the optimal solution in four instances. Secondly, as can be seen in Table 1, the STA model demonstrated superior performance compared to the $M\&S$ model across most of the 5-group evaluation criteria (n/m), except for the initial category (ratio 2.0). STA presented an average percentage gain of approximately 400% in terms of time efficiency.

Additionally, tests were performed with the relaxed version of the models (STA_R and $M\&S_R$) to evaluate the gap between the optimal solution and the value obtained with the relaxation. The gap was calculated using the formula $\Omega\% = \frac{FO - FO_R}{FO} \times 100$, where FO represents the optimal value and FO_R the linear relaxation value. The comparative results are presented in Table 2, which displays the average of 10 instances for each instance category (n/m) separated by the respective class $C1 - C7$. It can be seen that the $M\&S_R$ model exhibits higher bounds for all classes with ratio (2.0), as well as in class $C5$ in the groups with ratio $(2.25, 2.5)$. The STA_R model obtained better bounds in classes $C3$, $C4$ and $C7$ in the groups with ratio $(2.25, 2.5, 2.75)$ and in class $C6$ in the groups with ratio $(2.25, 2.5)$. In both models, the percentage difference between the optimal and relaxed values was minimal in the instances with a ratio (3.0) in all the categories analyzed.

Table 3 provides comprehensive information about the tests performed, presenting the average resolution time in seconds for each pair's set of 10 instances (n/m). In this table, it is possible to identify four instances that were not optimally solved with the $M\&S$ model, one from the arrangement $(200/80)$ and the other three from the arrangement $(198/72)$. The results can be compared for specific job and machine configurations in the seven classes identified as *C1-C7*. It is worth noting that the STA model shows superior performance, especially in arrangements with more than 100 jobs. A significant reduction in resolution time is observed in classes $C6$ and $C7$.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

n	m	$M\&S$ (Ω%)							STA (Ω%)						
		C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*
							n/m = 2.0								
20	10	**1.03**	**1.63**	**1.38**	**0.00**	**2.17**	**1.52**	**1.83**	4.08	3.75	3.08	3.11	6.13	4.46	3.43
40	20	**1.42**	**2.87**	**0.20**	**2.74**	**0.47**	**1.51**	**1.61**	5.35	3.78	1.75	2.89	7.06	3.35	2.69
60	30	**1.00**	3.53	**1.21**	**2.31**	**1.02**	**2.56**	**0.88**	4.36	**2.97**	1.79	2.52	7.02	2.85	2.43
80	40	**1.74**	**1.66**	**0.71**	**2.06**	**0.78**	**1.80**	**1.78**	4.35	2.64	2.12	2.54	8.96	2.49	2.26
100	50	**0.77**	**1.50**	**1.11**	**0.88**	**0.00**	**1.26**	**0.78**	4.33	2.34	1.44	1.66	10.02	2.56	2.34
120	60	**2.35**	**1.48**	**1.24**	1.57	**0.00**	**1.45**	2.09	3.65	2.28	1.70	**1.42**	10.20	2.62	**1.88**
140	70	**1.87**	**1.38**	1.77	2.32	**0.00**	**1.07**	**1.85**	3.83	1.84	**1.71**	**1.98**	13.32	2.14	1.95
160	80	**1.22**	**1.65**	**1.31**	1.81	**0.24**	**1.10**	**1.02**	3.06	2.47	1.38	1.81	10.12	2.20	1.77
180	90	**1.29**	**1.46**	**1.18**	1.57	**0.00**	**0.43**	**1.60**	2.98	1.95	1.31	**1.42**	13.80	2.13	1.76
200	100	**2.49**	**1.70**	**0.86**	1.81	**0.00**	**1.27**	**1.51**	3.80	1.86	1.38	**1.52**	16.34	1.93	1.68
avg		**1.52**	**1.89**	**1.10**	**1.71**	**0.47**	**1.40**	**1.50**	3.98	2.59	1.77	2.09	10.30	2.67	2.22
							n/m = 2.25								
36	16	**1.52**	1.84	3.82	7.45	1.55	1.61	3.64	3.31	**1.35**	**2.94**	**5.68**	**1.23**	**1.32**	**2.64**
54	24	**2.09**	1.73	4.08	6.29	**0.89**	1.44	3.25	3.50	**1.39**	**3.18**	**4.54**	2.95	**1.05**	**2.33**
72	32	**1.00**	1.18	4.20	6.07	**0.70**	1.18	2.57	2.73	**1.11**	**3.19**	**4.57**	3.73	**0.86**	**1.77**
90	40	**1.83**	1.16	2.88	6.34	**0.68**	0.98	1.43	2.89	**1.03**	**2.24**	**4.73**	1.99	**0.78**	**1.06**
108	48	**1.19**	0.59	3.60	5.45	**0.35**	1.06	1.52	2.22	0.59	**2.70**	**3.93**	4.78	**0.78**	**0.98**
126	56	**2.13**	0.72	2.42	5.44	**0.26**	0.85	1.39	2.33	0.72	**1.84**	**3.86**	4.74	**0.68**	**0.86**
144	64	**1.55**	0.79	3.60	5.38	**0.22**	0.82	1.87	1.57	**0.65**	**2.63**	**3.85**	2.91	**0.61**	**1.23**
162	72	**1.86**	0.51	3.26	5.19	**0.04**	0.80	1.24	1.95	0.51	**2.46**	**3.71**	4.46	**0.66**	**0.78**
180	80	**3.03**	0.51	3.61	5.12	**0.09**	0.65	1.21	3.25	0.51	**2.71**	**3.63**	5.63	**0.48**	**0.73**
198	88	1.47	0.29	3.09	4.64	**0.13**	0.62	1.52	**1.38**	0.29	**2.35**	**3.29**	5.43	**0.45**	**0.93**
avg		**1.77**	0.93	3.46	5.74	**0.49**	1.00	1.96	2.51	**0.81**	**2.62**	**4.18**	3.78	**0.77**	**1.33**
							n/m = 2.5								
20	8	**0.85**	1.22	4.29	5.59	**1.80**	1.77	3.42	1.80	**1.01**	**2.63**	**3.19**	2.27	**1.29**	**2.08**
40	16	**1.18**	0.78	4.08	6.27	**0.59**	0.85	3.26	1.78	0.78	**2.20**	**3.44**	0.66	**0.78**	**1.86**
60	24	0.41	0.59	3.62	4.88	0.56	0.50	1.54	0.41	**0.52**	**2.06**	**2.52**	0.56	**0.42**	**0.74**
80	32	0.22	0.20	3.80	4.80	0.20	0.51	2.05	0.22	0.20	**2.03**	**2.59**	0.20	**0.43**	**0.98**
100	40	0.52	0.27	3.92	4.79	**0.12**	0.39	1.63	0.52	0.27	**1.96**	**2.47**	1.76	**0.31**	**0.81**
120	48	**0.00**	0.13	3.93	4.58	0.08	0.19	1.80	0.40	0.13	**2.04**	**2.37**	0.08	**0.15**	**0.89**
140	56	0.16	0.07	3.45	4.67	0.04	0.16	1.59	0.16	0.07	**1.91**	**2.48**	0.04	**0.14**	**0.73**
160	64	0.00	0.00	3.89	5.19	**0.00**	0.13	1.87	0.00	0.00	**2.10**	**2.75**	0.24	0.13	**0.87**
180	72	0.32	0.06	3.44	5.02	**0.04**	0.11	1.49	0.32	0.06	**1.80**	**2.66**	1.08	**0.10**	**0.66**
200	80	0.47	0.00	3.53	4.95	**0.04**	0.07	1.56	0.47	0.00	**1.84**	**2.61**	2.00	0.07	**0.66**
avg		**0.41**	0.33	3.80	5.08	**0.35**	0.47	2.02	0.61	**0.30**	**2.06**	**2.71**	0.89	**0.38**	**1.03**
							n/m = 2.75								
22	8	**1.19**	0.57	2.69	2.67	**0.68**	0.64	1.50	1.62	0.57	**1.12**	**1.05**	1.30	**0.57**	**0.72**
44	16	0.14	0.06	2.42	1.83	0.10	0.27	0.98	0.14	0.06	**0.74**	**0.58**	0.10	0.27	**0.41**
66	24	0.00	0.00	2.54	2.19	0.00	0.11	0.73	0.00	0.00	**0.87**	**0.74**	0.00	0.11	**0.25**
88	32	0.07	0.00	2.22	1.99	**0.00**	0.05	0.90	0.07	0.00	**0.76**	**0.60**	0.85	0.05	**0.29**
110	40	0.21	0.00	2.36	2.56	0.00	0.01	0.58	0.21	0.00	**0.85**	**0.77**	0.00	0.01	**0.21**
132	48	0.00	0.00	2.27	2.55	0.00	0.00	0.62	0.00	0.00	**0.66**	**0.74**	0.00	0.00	**0.18**
154	56	0.00	0.00	2.36	2.17	**0.00**	0.00	0.67	0.00	0.00	**0.80**	**0.61**	0.21	0.00	**0.17**
176	64	0.00	0.00	2.27	2.01	0.00	0.00	0.71	0.00	0.00	**0.80**	**0.67**	0.00	0.00	**0.20**
198	72	0.00	0.00	2.14	2.13	0.00	0.02	0.76	0.00	0.00	**0.67**	**0.71**	0.00	0.02	**0.19**
220	80	0.00	0.00	2.58	2.08	**0.00**	0.01	0.50	0.00	0.00	**0.89**	**0.67**	0.30	0.01	**0.14**
avg		**0.16**	0.06	2.38	2.22	**0.08**	0.11	0.79	0.20	0.06	**0.82**	**0.71**	0.28	**0.10**	**0.28**
							n/m = 3.0								
36	12	0.18	0.06	0.09	0.03	0.23	0.08	0.09	0.18	0.06	0.09	0.03	0.23	0.08	0.09
54	18	0.14	0.00	0.14	0.00	0.00	0.00	0.01	0.14	0.00	**0.09**	0.00	0.00	0.00	0.01
72	24	0.00	0.00	0.04	0.00	0.00	0.02	0.00	0.00	0.00	0.04	0.00	0.00	0.02	0.00
90	30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
108	36	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
126	42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
144	48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
162	54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
180	60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
198	66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
avg		0.03	0.01	0.03	0.00	0.02	0.01	0.01	0.03	0.01	**0.02**	0.00	0.02	0.01	0.01

Table 2: Average difference in optimal values between $M\&S$ and STA models and their relaxed versions.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

n	m	$M\&S$ (seconds)							STA (seconds)						
		C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*
							n/m = 2.0								
20	10	**0.04**	**0.04**	**0.02**	**0.02**	**0.04**	**0.03**	**0.03**	0.08	0.06	0.04	0.04	0.18	0.05	0.05
40	20	**0.05**	**0.05**	**0.03**	**0.05**	**0.08**	**0.06**	**0.11**	0.35	0.10	0.05	0.06	1.29	0.12	0.18
60	30	**0.08**	**0.10**	**0.04**	**0.07**	**0.13**	**0.16**	**0.10**	0.52	0.20	0.05	0.09	2.18	0.41	0.40
80	40	**0.14**	**0.20**	0.05	**0.10**	**0.11**	**0.27**	**0.20**	0.59	0.29	0.05	0.13	2.11	1.12	1.35
100	50	**0.13**	**0.19**	0.07	0.19	**0.16**	**0.55**	**0.31**	0.46	0.31	**0.06**	**0.16**	1.50	1.23	1.29
120	60	**0.30**	0.20	0.09	**0.14**	**0.12**	**0.37**	1.49	0.64	**0.18**	**0.06**	0.19	1.38	0.63	**1.36**
140	70	**0.30**	0.33	0.10	0.30	**0.16**	**0.58**	1.49	0.39	**0.26**	**0.06**	**0.24**	0.86	1.00	**1.21**
160	80	**0.38**	0.21	0.12	**0.31**	**0.25**	**1.12**	3.25	0.52	**0.15**	**0.05**	0.34	0.86	1.76	**1.57**
180	90	**0.29**	0.24	0.08	**0.22**	**0.26**	**0.72**	**2.75**	0.38	**0.16**	**0.05**	0.31	0.88	1.03	2.78
200	100	**0.31**	0.34	0.10	0.37	**0.31**	1.87	3.26	0.56	**0.18**	**0.05**	**0.27**	0.90	**1.50**	**2.36**
avg		**0.20**	0.19	0.07	0.18	**0.16**	0.57	1.30	0.45	0.19	**0.05**	0.18	1.21	0.89	**1.26**
							n/m = 2.25								
36	16	**0.12**	**0.25**	0.33	**0.37**	**0.32**	**0.34**	**0.46**	0.55	0.39	**0.21**	0.44	1.72	0.48	1.16
54	24	**0.14**	**0.39**	0.59	**0.75**	**0.38**	**1.00**	**1.46**	0.71	0.57	**0.30**	0.77	3.03	2.00	2.85
72	32	**0.24**	0.67	0.95	1.26	**0.61**	1.92	4.02	0.93	**0.64**	**0.45**	**1.06**	2.66	**0.98**	**1.66**
90	40	**0.27**	0.56	1.24	2.12	**1.39**	5.93	9.46	0.81	**0.42**	**0.39**	**1.13**	1.51	**1.07**	**1.84**
108	48	**0.28**	0.90	1.62	2.90	1.83	12.42	20.67	0.68	**0.36**	**0.41**	**1.17**	**0.95**	**1.85**	**3.42**
126	56	**0.50**	1.31	2.06	4.32	2.17	16.61	37.05	0.85	**0.50**	**0.40**	**1.34**	**1.30**	**2.45**	**5.65**
144	64	**0.48**	2.19	2.56	5.45	2.19	38.00	72.84	0.62	**0.54**	**0.40**	**0.98**	**0.98**	**12.14**	**6.64**
162	72	0.90	1.73	3.37	6.08	1.53	43.07	108.56	**0.69**	**0.42**	**0.36**	**0.93**	**0.88**	**9.05**	**12.09**
180	80	**0.45**	2.67	3.95	8.20	1.88	68.58	164.18	0.65	**0.41**	**0.35**	**1.22**	**0.76**	**19.12**	**17.81**
198	88	1.43	1.86	4.06	10.42	3.75	104.31	338.00	**0.52**	**0.34**	**0.33**	**1.07**	**0.87**	**11.59**	**36.80**
avg		**0.48**	1.25	2.07	4.19	1.61	29.22	75.67	0.70	**0.46**	**0.36**	**1.01**	**1.47**	**6.07**	**8.99**
							n/m = 2.5								
20	8	**0.09**	**0.11**	0.14	**0.17**	**0.15**	0.12	**0.16**	0.22	0.16	0.14	0.22	0.55	**0.09**	0.17
40	16	**0.19**	**0.34**	0.43	**0.56**	**0.62**	**0.56**	**0.98**	0.97	0.75	**0.31**	0.68	3.36	1.25	2.45
60	24	**0.24**	**0.55**	0.81	1.28	**1.20**	**1.53**	2.93	1.01	0.76	**0.52**	**1.22**	2.40	1.93	**2.41**
80	32	**0.43**	**0.60**	1.31	1.77	2.08	6.54	11.85	1.30	0.63	**0.54**	**1.61**	**1.53**	**1.32**	**2.84**
100	40	**0.75**	0.85	1.60	3.68	2.18	11.13	25.46	1.59	**0.66**	**0.60**	**1.39**	**1.38**	**2.20**	**5.35**
120	48	**0.52**	1.15	2.65	5.39	1.98	18.65	48.99	0.81	**0.65**	**0.58**	**1.41**	**0.83**	**3.38**	**9.34**
140	56	0.97	1.05	3.59	8.81	2.87	47.30	100.53	**0.79**	**0.45**	**0.47**	**1.57**	**1.19**	**6.50**	**13.90**
160	64	1.02	1.06	4.79	12.14	2.39	71.65	193.30	**0.56**	**0.42**	**0.47**	**1.46**	**1.09**	**6.93**	**20.51**
180	72	1.25	1.77	6.86	11.18	3.87	139.88	379.34	**0.57**	**0.50**	**0.52**	**1.52**	**1.18**	**20.99**	**48.06**
200	80	1.19	1.33	6.64	15.28	5.15	200.47[*1]	592.64	**0.44**	**0.58**	**0.48**	**1.75**	**1.30**	**26.84**	**91.23**
avg		**0.67**	0.88	2.88	6.03	2.25	49.78[*1]	135.62	0.83	**0.56**	**0.46**	**1.28**	**1.48**	**7.14**	**19.63**
							n/m = 2.75								
22	8	**0.11**	**0.14**	0.16	**0.22**	**0.22**	**0.12**	**0.19**	0.54	0.26	0.16	0.31	1.44	0.20	0.26
44	16	**0.20**	**0.27**	0.45	**0.66**	**0.58**	**0.64**	**1.28**	1.81	1.00	**0.43**	0.96	3.53	2.36	3.84
66	24	**0.37**	**0.41**	**0.93**	1.63	**0.69**	**1.34**	5.15	1.60	0.99	0.65	**1.56**	1.59	1.39	**2.68**
88	32	**0.60**	**0.52**	1.30	3.14	**0.94**	3.94	24.12	1.39	0.78	**0.53**	**1.53**	1.43	**1.31**	**3.37**
110	40	**0.83**	0.83	1.75	4.46	1.35	9.85	59.05	1.14	**0.66**	**0.46**	**1.40**	**1.34**	**2.47**	**7.10**
132	48	0.95	0.93	2.79	6.04	3.09	7.47	126.69	**0.64**	**0.59**	**0.49**	**1.56**	**1.51**	**5.23**	**14.73**
154	56	1.43	1.16	3.78	8.69	3.52	**11.26**	270.18	**0.68**	**0.59**	**0.53**	**1.32**	**1.49**	11.81	**22.96**
176	64	1.66	1.37	4.50	6.36	4.67	53.91	555.76	**0.67**	**0.61**	**0.52**	**1.61**	**1.47**	**19.82**	**45.52**
198	72	1.83	1.84	4.53	10.42	6.91	238.77	931.26	**0.55**	**0.57**	**0.49**	**1.47**	**1.74**	**24.08**	**84.17**
220	80	2.00	1.77	6.26	9.66	6.67	231.71	1403.58[*3]	**0.54**	**0.63**	**0.50**	**1.87**	**1.80**	**45.97**	**143.49**
avg		1.00	0.92	2.65	5.13	2.87	55.90	337.73[*3]	**0.96**	**0.67**	**0.48**	**1.36**	**1.73**	**11.46**	**32.81**
							n/m = 3.0								
36	12	**0.22**	**0.23**	**0.15**	**0.20**	**0.54**	**0.28**	**0.29**	2.19	1.17	0.21	0.48	3.79	1.89	2.19
54	18	**0.31**	**0.30**	**0.23**	**0.32**	**0.73**	**0.71**	**0.90**	1.87	1.04	0.35	1.09	1.64	1.49	1.52
72	24	0.50	0.48	0.32	0.41	1.19	1.41	1.53	**1.51**	**1.13**	**0.32**	**0.94**	**1.49**	**1.24**	**1.34**
90	30	**0.63**	**0.66**	0.43	**0.69**	**1.39**	3.65	3.39	1.42	0.70	**0.34**	1.26	1.59	**2.30**	**2.71**
108	36	**1.11**	0.88	0.45	**0.74**	2.11	4.02	7.37	1.15	**0.68**	**0.35**	0.90	**1.34**	**3.56**	**6.27**
126	42	1.11	1.14	0.50	0.98	2.42	11.39	19.76	**1.05**	**0.71**	**0.23**	**0.82**	**1.62**	**4.78**	**13.87**
144	48	1.72	1.25	0.73	1.18	3.95	24.88	18.23	**1.01**	**0.69**	**0.39**	**0.97**	**2.18**	**10.81**	**18.99**
162	54	1.54	1.40	0.92	1.46	5.63	52.31	50.58	**0.73**	**0.78**	**0.26**	**1.10**	**2.31**	**25.25**	**25.40**
180	60	1.98	1.48	1.02	2.07	5.64	47.14	103.91	**0.61**	**0.68**	**0.21**	**1.47**	**2.60**	**32.90**	**76.40**
198	66	2.41	1.82	1.00	2.10	7.50	116.01	178.44	**0.58**	**0.84**	**0.22**	**1.38**	**1.89**	**45.20**	**74.03**
avg		**1.15**	0.97	0.57	**1.02**	3.11	26.18	38.44	1.21	**0.84**	**0.29**	1.04	**2.04**	**12.94**	**22.27**

[*] number of instances with times discarded from the average calculation for not proving optimality

Table 3: Detailed computational results for the $M\&S$ and STA models.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

Conclusions and Future Work

This paper presents a mathematical model based on arc flow to address the $P||Cmax$ problem, representing an improvement over the model proposed by Mrad and Souayah [2018]. Symmetry-breaking strategies were adopted, and a new approach to generating the graph was provided to reduce the size of the generated graph and the number of variables used.

Tests were performed on 3500 instances generated as specified in the literature, for which the proposed model showed an average percentage time gain of approximately 400% compared to the literature model. For the first time, we have successfully demonstrated optimal results across all instances of this type. Notably, the instances that obtained the best comparative results are those with more than 100 jobs belonging to classes $C6$ and $C7$.

These classes of instances were listed for investigation in future work during the research conducted by the authors of the reference model Mrad and Souayah [2018]. All computational testing was conducted using the CPLEX 20.1 with the default configuration. Comparative results are also presented regarding the lower bound provided by the linear relaxation of both models.

The present methodology has a clear potential that can be exploited in applications to similar problems. In future work, we intend to present computational results for the job scheduling problem on identical parallel machines with sequence-dependent times, as well as the analysis of a set of 780 difficult instances described in the work of Dell'Amico et al. [2008].

References

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

da Silva, H. V., Lemos, F. K., Cherri, A. C., and de Araujo, S. A. (2023). Arc-flow formulations for the one-dimensional cutting stock problem with multiple manufacturing modes. *RAIRO Operations Research*, 57(1):183–200.

Dell'Amico, M., Iori, M., Martello, S., and Monaci, M. (2008). Heuristic and exact algorithms for the identical parallel machine scheduling problem. *INFORMS Journal on Computing*, 20(3): 333–344.

Dell'Amico, M. and Martello, S. (1995). Optimal scheduling of tasks on identical parallel processors. *ORSA Journal on Computing*, 7(2):191–200.

Ford, L. R. and Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433.

Garey, M. R. and Johnson, D. S. (1978). "Strong" np-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508.

Gharbi, A. and Bamatraf, K. (2022). An improved arc flow model with enhanced bounds for minimizing the makespan in identical parallel machine scheduling. *Processes*, 10(11).

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17:416–429.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5(2):287–326.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118

Hoffmann, K. and Buscher, U. (2019). Valid inequalities for the arc flow formulation of the railway crew scheduling problem with attendance rates. *Comput. Ind. Eng.*, 127:1143–1152.

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

Laha, D. (2012). A simulated annealing heuristic for minimizing makespan in parallel machine scheduling. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, volume 7677 LNCS, p. 198–205. Springer.

Laha, D. and Gupta, J. N. (2018). An improved cuckoo search algorithm for scheduling jobs on identical parallel machines. *Computers & Industrial Engineering*, 126:348–360.

Macedo, R., Alves, C., and Valério de Carvalho, J. M. (2010). Arc-flow model for the two dimensional guillotine cutting stock problem. *Computers & Operations Research*, 37(6):991–1001.

Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., and Strasdat, N. (2020). Improved flow-based formulations for the skiving stock problem. *Computers & Operations Research*, 113: 104770.

Martinovic, J., Scheithauer, G., and Valério de Carvalho, J. M. (2018). 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.

Mokotoff, E. (2004). An exact algorithm for the identical parallel machine scheduling problem. *European Journal of Operational Research*, 152:758–769.

Mrad, M. and Souayah, N. (2018). An arc-flow model for the makespan minimization problem on identical parallel machines. *IEEE Access*, 6:5300–5307.

Mrad, M. (2015). An arc flow-based optimization approach for the two-stage guillotine strip cutting problem. *Journal of the Operational Research Society*, 66(11):1850 – 1859.

Ramos, B., Alves, C., and Valério de Carvalho, J. M. (2022). An arc flow formulation to the multitrip production, inventory, distribution, and routing problem with time windows. *International Transactions in Operational Research*, 29(1):526 – 553.

Shapiro, J. F. (1968). Dynamic programming algorithms for the integer programming problem-i: The integer programming problem viewed as a knapsack type problem. *Operations Research*, 16:103–121.

Silva, J. M. P., Teixeira, E., and Subramanian, A. (2021). Exact and metaheuristic approaches for identical parallel machine scheduling with a common server and sequence-dependent setup times. *Journal of the Operational Research Society*, 72(2):444–457.

Trindade, R. S., de Araújo, O. C. B., and Fampa, M. (2020). Arc-Flow Approach for Parallel Batch Processing Machine Scheduling with Non-identical Job Sizes. In *International Symposium on Combinatorial Optimization*, volume 12176 LNCS, p. 179–190. Springer.

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

Wolsey, L. A. (1977). Valid inequalities, covering problems and discrete dynamic programs. *Annals of Discrete Mathematics*, 1:527–538.

DOI: 10.59254/sbpo-2023-175118
http://live-proceedings.pantheonsite.io/p/175118