Efficient arc-flow formulations for makespan minimisation on parallel machines with a common server

Alessandro Druetto, Andrea Grosso, Jully Jeunet, Fabio Salassa

Efficient arc-flow formulations for makespan minimisation on parallel machines with a common server

Alessandro Druetto[1], Andrea Grosso[1], Jully Jeunet[2]*, Fabio Salassa[3]

[1]Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, Torino 10149, Italy

[2]CNRS, Université Paris Dauphine, PSL Research University, CNRS UMR[7243], Lamsade, place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France

[3]Dipartimento di Ingegneria Gestionale e della Produzione (DIGEP), Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy

Abstract

We consider the problem of scheduling non preemptively a set of jobs on parallel identical machines with prior setup operations on a single shared server, where the objective is to minimise the makespan. We develop an arc-flow formulation to the problem with two multigraphs, one for the machines and one for the server, with a same set of nodes representing points in time, and arcs associated with job execution, and with machines or server idleness. The resulting formulation, called Flow-Flow formulation (FFF), and its tuned version (FFT) are compared with the best existing model in the literature, a time-indexed variable formulation ($\mathcal{F}2$), on benchmark instances with up to 200 jobs and 10 machines. Computational results showed that our Flow-Flow models outperformed $\mathcal{F}2$ especially for instances with more than 50 jobs and optimally solved a majority of problems with 150 and 200 jobs for which $\mathcal{F}2$ found only very few optimal solutions.

Keywords: Scheduling; arc-flow formulation; common server; parallel machines; makespan minimisation

1 Introduction

In many manufacturing systems, a common server such as a human operator, a robot or a tool needs to be shared by a number of parallel machines to implement setups or loads. Scheduling problems with a single server occur frequently in automated material handling systems; in flexible manufacturing systems where an automated guided vehicle is used to load jobs on machines (Hall et al., 2000); in the printing industry where a team of workers must clean and reset presses each time a new order is received (Huang et al., 2010) or similarly in knitted fabrics with knitting machines needing to be emptied and their needles, repositioned (Kerkhove and Vanhoucke, 2014). Sharing the server resource results in machine idle time that can be reduced or eliminated by developing a good schedule.

The focus of this paper is on the identical parallel machine scheduling problem with sequence independent setup times and a single shared server where the objective is to minimise the makespan. The problem denoted as $P, S1|s_j, p_j|C_{\max}$ is to schedule a set of jobs $N = \{1, 2, \ldots, n\}$ on an arbitrary number m of identical parallel machines ($m \geq 2$), where each job $j \in N$ must be processed non preemptively on one of the machines for p_j units of time. Prior to processing, s_j units of time must be spent for setup operations on a single shared server. Since the server is shared among all machines, no more than one setup operation can take place at each point in time.

Seminal contributions to the problem mostly consider two machines with equal processing or setup times, and provide complexity analyses and solution methods (Koulamas, 1996; Kravchenko and Werner, 1997, 2001; Glass et al., 2000; Hall et al., 2000 among others). For an extended and clear survey that

*Corresponding author. E-mail addresses: alessandro.druetto@unito.it, andrea.grosso@unito.it, jully.jeunet@dauphine.fr, fabio.salassa@polito.it.

uses the standard three-field notation described by Graham et al. (1979), the reader may refer to Bektur and Sarac (2019).

The problem with an arbitrary number of machines $m \geq 2$ and general job sets was first optimally addressed by Kim and Lee (2012) who provided two Mixed Integer Programming (MIP) formulations. The first one uses the sequence of setups of the server whereas the second one relies on the server waiting time. Experiments from Koulamas (1996) were adapted to create server waiting time when there is more than 2 machines. Instances with $n = 10$ jobs and 2 to 4 machines were exactly solved in less than one minute. For larger problems up to 40 jobs and 6 machines, 40 to 70% of optimal solutions on average were found within the limited runtime of 3600 s. Elidrissi et al. (2018b) addressed the same scheduling problem for which they proposed two MIP formulations based on completion time variables and time-indexed variables with better performance than the MIP formulations of Kim and Lee (2012). The authors then enhanced these formulations with strengthening constraints (Elidrissi et al., 2021). In addition, they developed three other MIP formulations using respectively network variables, linear ordering variables and completion time variables. The performance of these models was compared with the two MIP formulations of Kim and Lee (2012) on a set of instances generated in a similar way. Results showed that only the time-indexed variable formulation (TIV I) was able to find optimal solutions to some of the instances with more than 10 jobs and up to 100 jobs. Quite recently, Silva et al. (2023) proposed several time indexed formulations among which $\mathcal{F}2$ performed substantially better than TIV I on an extended set of instances up to 100 jobs.

Metaheuristic solution approaches to the problem are available in Kim and Lee (2012) who developed a Simulated Annealing (SA) algorithm combined with tabu search, and in Elidrissi et al. (2020) who proposed a Variable Neighbourhood Search (VNS) algorithm. Elidrissi et al. (2018a) generalised the heuristics of Abdekhodaee and Wirth (2002) and that of Hasani et al. (2016) to the case of an arbitrary number of machines, thus providing two greedy heuristics aimed at minimising machine idle time and server waiting time.

Besides, several variants of the parallel machine scheduling problem with common servers have been considered in the literature, depending upon the objective to minimise, the type of setup times, the assumption about preemption or the number of servers. Changing only one assumption of the problem considered in this paper, we highlight the following contributions.

Liu et al. (2019) developed a branch-and-bound algorithm to minimise the weighted job completion time which was able to optimally solve instances with no more than 20 jobs and 3 machines. Abu-Shams et al. (2022) proposed a heuristic-based Genetic Algorithm (GA) to minimise tardiness so as to deal with large-sized problems up to 2000 jobs and 10 machines.

Sequence-dependent setup times are considered in Hamzadayi and Yildiz (2017) who presented a mixed integer linear programming (MILP) model for small-sized instances (no more than 20 jobs and 5 machines) as well as a SA and a GA for larger problems (up to 100 jobs and 10 machines). For the same problem, Silva et al. (2021) proposed an arc-time-indexed formulation which outperformed the MILP model of Hamzadayi and Yildiz (2017) on a set of instances with up to 21 jobs and 7 machines. For larger instances, neither of the two models was able to find optimal solutions.

Cheng et al. (2017) presented complexity analyses to the problem with preemption whereas Elidrissi et al. (2022) provided a MIP model to deal with 2 servers and small problems (10 jobs, 5 machines) as well as a VNS for large instances (250 jobs, 5 machines), but for regular jobs only (regular jobs are such that $p_i \leq p_j + s_j,\ \forall(i,j)$).

We propose an arc-flow formulation for problem $P, S1|s_j, p_j|C_{\max}$ with two multigraphs, one for the machines and one for the server, with a same set of nodes representing points in time, and arcs associated with job execution, and with machines or server idleness. The resulting formulation, called Flow-Flow formulation (FFF) and its tuned version (FFT) are capable to outperform the time-indexed formulation ($\mathcal{F}2$) of Silva et al. (2023), in terms of computation time and number of optimal solutions, especially on instances with more than 100 jobs.

The remainder of this paper is organised as follows. Our Flow-Flow formulation is presented in Section 2. Results of the computational experiments are discussed in Section 3. Finally, concluding remarks and directions for future research are provided in Section 4.

2 Flow-Flow formulations and existing mathematical model $\mathcal{F}2$

We first provide in Section 2.1 the bounds on the optimal makespan since the same upper bound is used in both $\mathcal{F}2$ and our formulations. The proposed Flow-Flow formulation is presented in Section 2.2. Finally, in Section 2.3, we compare our formulation to the model of Silva et al. (2023), $\mathcal{F}2$, and we show that the continuous relaxation of our formulation provides a better lower bound than a trivial bound.

2.1 Upper and lower bounds on the makespan

Upper bound. To derive an upper bound on the optimal makespan, Elidrissi et al. (2021) used the two greedy heuristics HS1, HS2 they developed in Elidrissi et al. (2018a). Heuristic HS1 aims at minimising machines idle time whereas HS2 seeks to minimise the server idle time. Both heuristics are based on ordering jobs according to six priority rules, namely shortest processing time (SPT); longest processing time (LPT); shortest setup time (SST); longest setup time (LST); shortest completion time (SCT) and longest completion time (LCT). A numerical illustration is provided in the Appendix.

As many jobs can have identical processing or setup times, we introduce a tie breaking rule that consists of arranging jobs with same criterion value according to a second criterion consistent with the first one. For instance, if rule LPT is used, jobs with same processing times are secondarily ordered according to LST. Analogously, with SPT we apply SPT+SST. Finally for SCT, we use SCT+SPT. Again, the reader can refer to the illustration in the appendix.

The upper bound $\texttt{UB}$ on the optimal makespan simply consists of the minimum of the makespan values obtained from heuristics HS1 and HS2. The horizon length T is set equal to this upper bound both in our Flow-Flow models and $\mathcal{F}2$. Formally we have

$$T = \texttt{UB} = \min\left(C_{\max}^{\text{HS1}}, C_{\max}^{\text{HS2}}\right). \tag{1}$$

Lower bounds. A trivial lower bound $\texttt{LB}_{\text{PMTN}}$ on the optimal makespan $C_{\max}^{\text{opt}}$ is given by

$$\texttt{LB}_{\text{PMTN}} = \frac{1}{m}\sum_{j=1}^{n}(s_j + p_j). \tag{2}$$

As a better lower bound than $\texttt{LB}_{\text{PMTN}}$, we use that of Elidrissi et al. (2021) who state that (i) if there is no server waiting time in the optimal schedule, then $C_{\max}^{\text{opt}}$ is equal to the sum of setup times and the shortest processing time; (ii) if there is no machine idle time, then $C_{\max}^{\text{opt}}$ is equal to the average over the number of machines of the sum of all jobs execution times, and the weighted sum of the $(m-1)^{\text{th}}$ first jobs setup times ranked in increasing order $\{\sigma(j)\}_{j=1..n}$. Formally this lower bound, $\texttt{LB}_{\text{Improved}}$, is given by

$$\texttt{LB}_{\text{Improved}} = \max\left(\sum_{j=1..n} s_j + \min_{1\leq j\leq n} p_j, \texttt{LB}_{\text{PMTN}} + \frac{\sum_{j=1}^{m-1}(m-j)s_{\sigma(j)}}{m}\right). \tag{3}$$

This improved lower bound is used in the tuned version of our Flow-Flow formulation as explained further in Section 3.2. The trivial lower bound in Eq. (2) is considered when we compare in Section 2.3 the existing model $\mathcal{F}2$ to our formulation.

2.2 Flow-Flow Formulation (FFF)

Arc-flow formulations allow for the use of a pseudo-polynomial number of variables and constraints, and have been recently applied to classical optimisation problems such as the cutting-stock problem (Martinovic et al., 2018), the bin-packing problem (Brandao et al., 2016) or the berth allocation problem (Kramer et al., 2019b). In the area of scheduling and most closely related to our problem, Mrad and Souayah (2018) proposed an arc-flow formulation for makespan minimisation on identical parallel machines and showed its efficiency to solve most of the hard instances from the literature. Gharbi and Bamatraf (2022) provided an improved arc-flow model for the same problem, with enhanced bounds. Results on benchmark instances with up to 200 jobs and 100 machines showed the superiority of their

model over that of Mrad and Souayah (2018). Kramer et al. (2019a) also considered the scheduling problem on identical parallel machines but with the aim of minimising the total weighted completion time. They developed enhanced arc-flow formulations able to solve exactly large-sized instances up to 400 jobs. This work was then extended in Kramer et al. (2020) to jobs with release dates.

For the problem with a common server under sequence dependent setup times, Silva et al. (2021) have proposed a quite sophisticated arc-time-indexed model with $O(n^2T)$ variables, allowing to solve exactly instances with up to 21 jobs and 7 machines. Under independent setup times, as in our case, an arc-flow formulation was recently developed by Silva et al. (2023) but suffers from bottleneck constraints. By contrast, our formulation has a reduced number of $O(nT)$ variables, eliminates the bottleneck constraints, and replaces them with a much stronger formulation involving a min-sum objective.

In this Section, we first provide the multigraph representation of the problem and we introduce the notations. The problem modelling is then illustrated with a scheduling example. Next, we present the mathematical formulation and the procedure to handle identical jobs.

2.2.1 Multigraph representation and notations

Our arc-flow formulation uses two multigraphs $G_{K\in\{M,S\}}(V, A_K)$ in order to model the scheduling of the m-machines collection, M, and the single server S, respectively. Our Flow-Flow formulation (FFF in the following), therefore contains an arc-flow formulation for both machines and server. The two graphs have the same set of nodes V, each node representing a unit time slot

$$V = \{0, 1, 2, \ldots, T\},$$

where T is the time horizon over which the set of jobs $N = \{1, 2, \ldots, n\}$ must be scheduled. As in $\mathcal{F}2$, we set T equal to the value of the upper bound on the makespan in Eq. (1).

The set A_K of arcs of multigraph $K \in \{M, S\}$ is defined as

$$\begin{array}{lcl} A_K & = & \{a_{jt}=(t,t+b_j;j)\colon j\in N, t=0..T-s_j-p_j\} \cup \{a_t=(t,t+1)\colon t=0..T-1\}, \\ \text{with } b_j & = & \begin{cases} s_j+p_j, & \text{if } K=M, \\ s_j, & \text{if } K=S. \end{cases} \end{array}$$

An arc a_{jt} represents the possible execution of job j in time interval $(t, t+b_j)$ so it links node t (start time) to node $t+b_j$ (end time) both on the machines graph G_M and on the server graph G_S. An arc a_t expresses the idleness of the server or machines in time interval $(t, t+1)$. In addition, we let $\delta^-_{[G_K,t]}$, $\delta^+_{[G_K,t]}$ be respectively the set of ingoing and outgoing arcs a_{jt} in/from node t in graph G_K, $K \in \{M, S\}$.

We let x_{jt} be a binary variable that takes a value of 1 if job j starts at time t and 0 otherwise. If $x_{jt} = 1$, this unit flow is placed on arc a_{jt} that links node t to node $t + s_j$ in graph G_S and node t to node $t + s_j + p_j$ in graph G_M. Job j thus starts at time t both on the server and on a machine since the non preemption assumption leads to reserve a machine for processing the job as soon as its setup starts on the server. Thus, the machine is not considered as idle on t if the setup on the server starts on t.

We let y^M_t, y^S_t be the integer variables expressing the number of idle machines and server respectively. Variable y^S_t is obviously binary since we consider a single server. The flow value y^M_t (resp. y^S_t) is placed on arc a_t in graph G_M (resp. G_S) that connects node t to node $t+1$.

Finally we introduce a binary variable z_t that takes a value of 1 if the last scheduled job ends at time t and 0 otherwise. If $z_t = 1$ the makespan is therefore equal to t.

Table 1 summarises the notations and definitions.

Indices	
j	Job
t, τ	Time
$K \in \{M, S\}$	Type of multigraph (M for the machines, S for the server)
Parameters	
T	Time horizon (set to the upper bound on the makespan in Eq. (1))
$N = \{1, 2, \ldots, n\}$	Set of jobs to be scheduled
s_j	Setup time of job j (on the server)
p_j	Processing time of job j (on a machine)
m	Number of machines
Definitions	
$G_K(V, A_K)$	Multigraph for $K \in \{M, S\}$
$V = \{0, 1, 2, \ldots, T\}$	Set of nodes shared by both multigraphs
$a_{jt} = (j; t, t+b_j)$	Execution arc of job j from t to $t + b_j$; $b_j = s_j$ in G_S, $b_j = s_j + p_j$ in G_M, $j \in N, \quad t = 0..T - s_j - p_j$.
$a_t = (t, t+1)$	Idleness arc from t to $t + 1$, $t = 0..T-1$
$A_K = \{a_{jt}\} \cup \{a_t\}$	set of arcs in the multigraph $K \in \{M, S\}$
$\delta^-_{[G_K,t]}$	set of ingoing execution arcs a_{jt} in node t in G_K, $K \in \{M, S\}$
$\delta^+_{[G_K,t]}$	set of outgoing execution arcs a_{jt} from node t in G_K, $K \in \{M, S\}$
Variables	
x_{jt}	Binary equal to 1 if job j starts at time t and 0 otherwise, $\forall j \in N$, $\forall t = 0..T - s_j - p_j$.
y^K_t	Number of idle machines (if $K = M$) or server (if $K = S$) at time t, $\forall t = 0..T$
z_t	Binary equal to 1 if the last scheduled job ends at time t, $\forall t = 1..T$

Table 1: Notations and definitions

2.2.2 Illustration of the problem modelling

To illustrate the problem modelling, let us consider $n = 5$ jobs with $\{s_1, \ldots, s_5\} = \{2, 3, 3, 2, 2\}$ and $\{p_1, \ldots, p_5\} = \{3, 5, 4, 5, 3\}$. The jobs are to be scheduled on a time horizon $T = 18$ with one server and $m = 3$ machines. Figure 1 displays on a same time scale the Gantt chart (upper part) of a feasible solution to the problem with $C_{\max} = 17$ and the associated flows on multigraphs G_M and G_S (lower part). The flows are generated at node 0 for both graphs. For instance, job 1 starts at time 0 ($x_{1,0} = 1$) so an execution arc with a unit flow is placed on G_S from node 0 to node $0+s_1 = 2$ and the corresponding nodes $\{0, 1, 2\}$ are connected with zero-flow (hence not drawn) idleness arcs since the single server is busy. On graph G_M an execution arc connects node 0 to node $0 + s_1 + p_1 = 5$ even if the processing of the jobs starts on M_1 only at time 2, as the machine is reserved as soon as the setup starts on the server. Thus, between time 0 and 5, M_1 is not idle; only machines M_2 and M_3 are idle between time 0 and 2, so idleness arcs with a flow of 2 units connect nodes $\{0, 1, 2\}$. At $t = 2$ job 2 starts its setup on the server, so M_2 is reserved and the idleness arc from node 2 to node 3 now carries a unit flow as only M_3 is idle. Let us note that in time interval $(8, 10)$ all the machines are busy, so the corresponding nodes are connected with zero-flow idleness arcs. All in all, 3 units of flow are routed on G_M, and 1 unit on G_S. The sink node is 17 as we have $z_{17} = 1$ so arcs are not drawn between nodes 17 and 18. The flow does not give explicitly an assignment jobs-machines, but it can be decomposed into 3 paths that correspond to machine schedules.

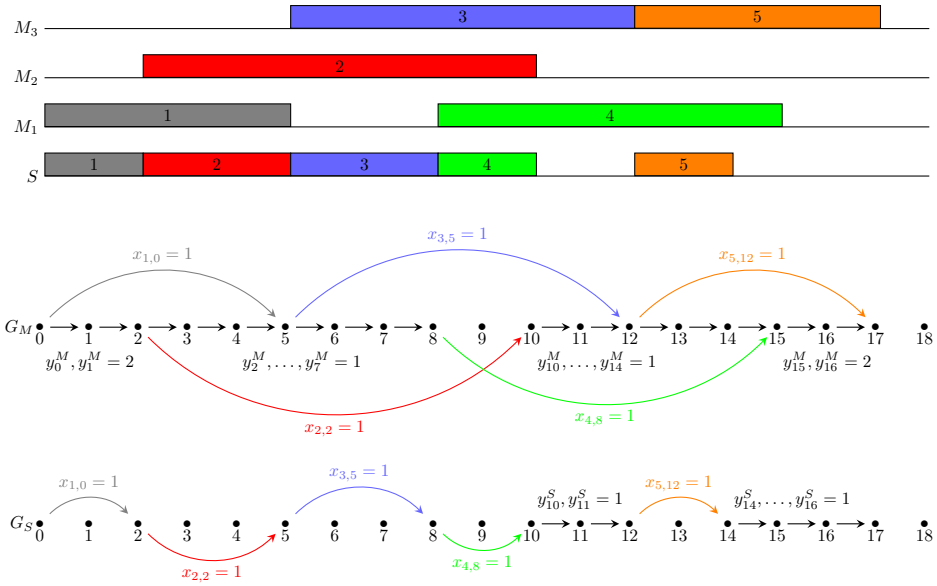

Figure 1: A scheduling example and corresponding flows on G_M and G_S.

2.2.3 Mathematical formulation (FFF) and identical jobs handling

In order to facilitate the reading of our formulation, we index all equations with time t and we indicate in parenthesis the values of t for which the corresponding constraints hold. Our arc-flow model (FFF) is written as follows

$$\min \sum_{t=1}^{T} t z_t \tag{4}$$

s.t.

$$\sum_{t=0}^{T-s_j-p_j} x_{jt} = 1 \qquad (j \in N) \tag{5}$$

$$\sum_{a_{jt} \in \delta^+_{[G_M,t]}} x_{jt} + y_t^M = m \qquad (t = 0) \tag{6}$$

$$\sum_{a_{jt} \in \delta^+_{[G_M,t]}} x_{jt} - \sum_{a_{j\tau} \in \delta^-_{[G_M,t]}} x_{j\tau} + y_t^M - y_{t-1}^M = -m z_t \qquad (t = 1..T-1) \tag{7}$$

$$- \sum_{a_{j\tau} \in \delta^-_{[G_M,t]}} x_{j\tau} - y_{t-1}^M = -m z_t \qquad (t = T) \tag{8}$$

$$\sum_{a_{jt} \in \delta^+_{[G_s,t]}} x_{jt} + y_t^S = 1 \qquad (t = 0) \tag{9}$$

$$\sum_{a_{jt} \in \delta^+_{[G_s,t]}} x_{jt} - \sum_{a_{j\tau} \in \delta^-_{[G_s,t]}} x_{j\tau} + y_t^S - y_{t-1}^S = -z_t \qquad (t = 1..T-1) \tag{10}$$

$$- \sum_{a_{j\tau} \in \delta^-_{[G_s,t]}} x_{j\tau} - y_{t-1}^S = -z_t \qquad (t = T) \tag{11}$$

$$x_{jt} \in \{0,1\} \qquad (j \in N, t = 0..T-s_j-p_j) \tag{12}$$

$$y_t^M \geq 0, \quad y_t^S \in \{0,1\} \qquad (t = 0..T-1) \tag{13}$$

$$z_t \in \{0,1\} \qquad (t = 1..T) \tag{14}$$

Constraints (5) require that a starting time is assigned to each job $j \in N$. Constraints (6)–(8) related to the machines make variables x_{jt}, y_t^M a set of flow variables, requiring m units of flow to be routed

on graph G_M from source node 0 to some sink node $t \in \{1, \ldots, T\}$ for which $z_t = 1$. Let us note that summing up constraints (6)–(8) we get $m(1 - \sum_{t=1}^{T} z_t) = 0$, so at most one z_t can take a value of 1.

Constraints (9)–(11) make variables x_{jt}, y_t^S a set of flow variables, requiring one unit of flow to be routed on graph G_S from node 0 to some node $t \in \{1, \ldots, T\}$ for which $z_t = 1$. These constraints guarantee that no more than one job can be processed on the server simultaneously. Given that $\sum_{t=1}^{T} z_t = 1$ must hold in all feasible solutions, the objective function (4) correctly represents the makespan.

In sets of jobs to be scheduled, it is not uncommon that quite large families of identical jobs — with identical setup and processing times — emerge. In such cases, a powerful optimisation procedure consists in keeping a single copy of identical jobs and schedule it a number of times equal to the number of copies. Formally, letting n_j be the number of jobs identical to j in set N, we keep only one copy of such a job j, and schedule it n_j times. We thus eliminate all duplicated jobs from N, so that no two jobs in N are identical; we compute the multiplicity n_j of each job j and replace constraints (5) with

$$\sum_{t=0}^{T-s_j-p_j} x_{jt} = n_j.$$

This procedure presents similarities with the identical job grouping adopted in Kramer et al. (2019a). In many instances the procedure can substantially reduce the number of variables involved in the model. Due to its simplicity we always apply it to the FFF model.

2.3 Comparison with the timed-indexed formulation of Silva et al. ($\mathcal{F}2$)

The time-indexed formulation of Silva et al. (2023), $\mathcal{F}2$, makes use of the same binary variables x_{jt}, and an auxiliary variable $C_{\max}$ so as to implement a min-max formulation. In our graph-based notation, the formulation writes out as follows.

$$\min C_{\max} \tag{15}$$

s.t.

$$\sum_{t=0}^{T-p_j} (t + s_j + p_j)\, x_{jt} \leq C_{\max} \qquad \forall j \in N \tag{16}$$

$$\sum_{t=0}^{T-s_j-p_j} x_{jt} = 1 \qquad \forall j \in N \tag{17}$$

$$\sum_{a_{jt} \in \delta^+_{[G_M,0]}} x_{j0} + y_0^M = m \tag{18}$$

$$\sum_{a_{jt} \in \delta^+_{[G_M,t]}} x_{jt} - \sum_{a_{j\tau} \in \delta^-_{[G_M,t]}} x_{j\tau} + y_t^M - y_{t-1}^M = 0 \qquad \forall t = 1..T-1 \tag{19}$$

$$- \sum_{a_{j\tau} \in \delta^-_{[G_M,T]}} x_{j\tau} - y_{T-1}^M = -m \tag{20}$$

$$\sum_{a_{jt} \in \delta^+_{[G_S,0]}} x_{j0} + y_0^S = 1 \tag{21}$$

$$\sum_{a_{jt} \in \delta^+_{[G_s,t]}} x_{jt} - \sum_{a_{j\tau} \in \delta^-_{[G_s,t]}} x_{j\tau} + y_t^S - y_{t-1}^S = 0 \qquad \forall t = 1..T-1 \tag{22}$$

$$- \sum_{a_{j\tau} \in \delta^-_{[G_S,T]}} x_{j\tau} - y_{T-1}^S = -1 \tag{23}$$

$$x_{jt} \in \{0,1\} \qquad \forall j \in N, \tag{24}$$

$$\forall t = 0..T - s_j - p_j \tag{25}$$

$$y_t^M \geq 0, \qquad y_t^S \in \{0,1\} \qquad \forall t = 0..T-1 \tag{26}$$

$$\tag{27}$$

$$C_{\max} \geq 0 \tag{28}$$

Constraints (16) force $C_{\max}$ to be equal to the completion time of the last scheduled job. In the $\mathcal{F}2$ model expressed by Eq. (15)–(28), the horizon T is set to the upper bound value on the optimal makespan provided by the heuristics of Elidrissi et al. (2018), as defined in Eq. (1).

Between the two models, FFF and $\mathcal{F}2$, we can highlight close relations and important differences. First of all, the procedure to handle identical jobs as described in Section 2.2.3 can be applied to FFF, but not to $\mathcal{F}2$ in its present form.

A key feature of our FFF model is the introduction of z_t variables for handling the makespan objective. By contrast, the bottleneck constraints (16) in $\mathcal{F}2$ severely affect the quality of the continuous relaxation of the model, as it frequently yields a lower bound below the trivial bound $\texttt{LB}_{\text{PMTN}}$ in Eq. (2). Empirically, for almost all instances tested in our numerical experiments, the continuous relaxation of $\mathcal{F}2$ produced lower bounds much smaller than the corresponding trivial bounds (see Section 3).

On the other hand, our Flow-Flow model FFF in Eq. (4)–(14) does not suffer from such an issue. Indeed, it exhibits the following property.

Property 1. The optimal value of the continuous relaxation of the Flow-Flow model (FFF) is never less than $\texttt{LB}_{\text{PMTN}}$.

Proof. We consider any feasible solution of the continuous relaxation and we prove that its objective function value cannot be less than the trivial lower bound. In order to ease the notation, we let $I_t = \sum_{a_{j\tau} \in \delta^-_{[G_M,t]}} x_{jt}$ and $O_t = \sum_{a_{jt} \in \delta^+_{[G_M,t]}} x_{jt}$, with $I_0 = 0$ and $O_T = 0$, so that constraints (6)–(8) can be rewritten, changing sign, as

$$\begin{aligned}
-m =& I_t - O_t && - y^M_t && (t = 0)\\
mz_t =& I_t - O_t && + y^M_{t-1} - y^M_t && (t = 1..T-1)\\
mz_t =& I_t - O_t && + y^M_{t-1} && (t = T).
\end{aligned}$$

We multiply each constraint by the corresponding t, sum them up and divide by m

$$\sum_{t=1}^{T} t z_t = \frac{1}{m}\sum_{t=0}^{T} t[I_t - O_t] + \frac{1}{m}\left[\sum_{t=1}^{T-1}(t y^M_{t-1} - t y^M_t) + T y^M_{T-1}\right].$$

We note the following.

i $\sum_{t=1}^{T} t z_t$ is precisely the objective function.

ii The sum $\sum_{t=0}^{T} t[I_t - O_t]$ is a weighted sum of all arc-flow variables x_{jt} of arcs a_{jt}. Hence, every x_{jt} will appear in the sum exactly twice: the first time in O_t, for the tail of arc a_{jt}, multiplied by $-t$, and the second time in $I_{t+s_j+p_j}$, for the head of the arc, multiplied by $(t + s_j + p_j)$. Hence the contribution of x_{jt} to the sum is $(s_j + p_j)x_{jt}$. Thus

$$\sum_{t=0}^{T} t[I_t - O_t] = \sum_{j=1}^{n} \sum_{t=0}^{T-s_j-p_j} (s_j + p_j) x_{jt}$$

and by constraint (5)

$$\sum_{t=0}^{T} t[I_t - O_t] = \sum_{j=1}^{n} \left[(s_j + p_j) \sum_{t=0}^{T-s_j-p_j} x_{jt}\right] = \sum_{j=1}^{n} (s_j + p_j).$$

iii For the sum involving the y_t variables,

$$\begin{aligned}
\sum_{t=1}^{T-1}(t y^M_{t-1} - t y^M_t) =& \sum_{t=1}^{T-1} t y^M_{t-1} - \sum_{t=1}^{T-1} t y^M_t =\\
=& \sum_{t=0}^{T-2} (t+1) y^M_t - \sum_{t=1}^{T-1} t y^M_t =\\
=& y^M_0 + \sum_{t=1}^{T-2} y^M_t - (T-1) y^M_{T-1}
\end{aligned}$$

hence

$$\frac{1}{m}\left[\sum_{t=1}^{T-1}(ty_{t-1}^{M}-ty_{t}^{M})+Ty_{T-1}^{M}\right]=\frac{1}{m}\sum_{t=0}^{T-1}y_{t}^{M}$$

By the above considerations, we get

$$\sum_{t=1}^{T}tz_{t}=\frac{1}{m}\sum_{j=1}^{n}(s_{j}+p_{j})+\frac{1}{m}\sum_{t=0}^{T-1}y_{t}^{M},$$

showing that the objective function is made of the trivial lower bound $\texttt{LB}_{\text{PMTN}}$ in Eq. (2) plus a nonnegative contribution from the idle variables y_t^M, thus proving the claim. □

This property establishes a minimum performance for our Flow-Flow Formulation but there is no proof of dominance of the lower bound from its continuous relaxation, $\texttt{LB}_{\text{FFF}}$, over that of $\mathcal{F}2$, $\texttt{LB}_{\mathcal{F}2}$. There may indeed be cases where $\texttt{LB}_{\mathcal{F}2}$ exceeds $\texttt{LB}_{\text{PMTN}}$ and $\texttt{LB}_{\text{FFF}}$ as illustrated in the following numerical example with $n = 5$ jobs; $m = 2$ machines; processing times $\{p_1, p_2, p_3, p_4, p_5\} = \{10, 15, 20, 25, 30\}$; setup times $\{s_1, s_2, s_3, s_4, s_5\} = \{60, 65, 70, 75, 80\}$. We obtain $\texttt{LB}_{\text{PMTN}} = 225$; $\texttt{LB}_{\text{FFF}} = 355 < \texttt{LB}_{\mathcal{F}2} = 360$. However, in practice, we expect processing times to be usually higher than set-up times.

3 Computational experiments

We first provide the benchmark instances. Next, we present the Tuned version of our Flow Flow Formulation, namely FFT. Results are then discussed, starting with problems up to $n = 100$ jobs and ending with the largest instances ($n = 150, 200$ jobs).

The exact models TIVI I, $\mathcal{F}2$, FFF and FFT were coded in C++ and linked with the CPLEX Callable Library (C API) version 22.1. Computational experiments were executed in Ubuntu Linux 22.04 under WSL2, on an Intel(R) Core(TM) i7-10700 CPU with 2.90GHz and 16GB of RAM machine, running Windows 11.

Additionally, we provide results in the Appendix comparing ATIF (Silva et al., 2021) with $\mathcal{F}2$, FFF, and FFT. This comparison shows that ATIF was unable to solve to optimality problems with more than 20 jobs, demonstrating its limitations relative to our flow flow formulation for sequence-independent setup times.

3.1 Benchmark instances

The first set of instances we used are those of Elidrissi et al. (2021) who considered regular job sets together with general ones. However we only included in our experiments general job sets since, as pointed by Kim and Lee (2012), regularity of all jobs to be scheduled rarely happens in practice. Following Kim and Lee (2012), Elidrissi et al. (2021) generated their instances using four parameters (n, m, α, ρ), with n between 10 and 100 jobs and m varying from 3 to 5 machines. Parameter α determines the interval from which processing times are uniformly drawn as integer values

$$p_j \sim U\left[(1-\alpha)E(p_j), (1+\alpha)E(p_j)\right], \quad E(p_j) = 25, \quad \alpha = \{0.1, 0.3, 0.5\}. \tag{29}$$

Obviously, the smaller α the higher the number of jobs with identical processing times.

Parameter ρ together with m determines the server load ρ/m. In order to create idle times for the server, setup times must be lower than processing times. This requirement is satisfied using $E(s_j) = (\rho/m) \cdot E(p_j)$ with $\rho \leq 1$. Setup times are therefore drawn as follows

$$s_j \sim U\left[(1-\alpha)(\rho/m)E(p_j), (1+\alpha)(\rho/m)E(p_j)\right], \quad \rho = \{0.5, 0.7, 1\}. \tag{30}$$

Again, a low α value leads to a high number of jobs with identical setup times.

As observed by Silva et al. (2023), the set of 230 instances of Elidrissi et al. (2021) was relatively easy to solve. Consequently Silva et al. (2023) used the method of Kim and Lee (2012) to generate additional

problems with $n = \{10, 20, 30, 50, 75, 100\}$; $m = \{2, 4, 6, 10\}$; $\alpha = \{0.1, 0.3, 0.5\}$ and $\rho = 1$. They made 5 replications for each combination of parameters with $m < n$, resulting in a total of 345 problems.

We considered large-sized instances with $n = 150, 200$, and similar to Silva et al. (2023), we made 5 replications for each combination of parameters, with $m = \{4, 6, 10\}$; $\alpha = \{0.1, 0.3, 0.5\}$ and $\rho = 1$. In this way, we generated 90 additional problems with $n > 100$. All the 665 tested problems are available at https://datacloud.di.unito.it/index.php/s/An5G9dpiWpqfzkL.

3.2 Tuned version of the Flow-Flow model (FFT)

The Flow-Flow Tuned model (FFT) is also defined by Eq. (4)–(14) as FFF, but includes several features that we list below.

Formulation strengthening. To strengthen the formulation, we use the lower bound $\texttt{LB}_{\text{Improved}}$ of Elidrissi et al. (2021) in Eq. (3) by setting to zero variables z_t in Eq. (13) for all $t < \texttt{LB}_{\text{Improved}}$. In this way, we guarantee that the lower bound of our MILP model, $\texttt{LB}_{\text{FFT}}$, is not worse than $\texttt{LB}_{\text{Improved}}$.

Branching priority and direction. Branching priority is enforced for variables x_{jt} during the branch-and-cut phase, with decreasing priority over the increase of t. The branching direction is set to up, so the up branch ($x_{jt} = 1$) is taken first at each node, since the aim is to set as soon as possible to 1 the variables close to the start of the sequence. In doing so, we obtain a kind of 'schedule from the beginning' way of branching, which usually makes sense in scheduling, instead of operating on some (almost) randomly selected fractional x_{jt} as is the case with a default setting.

CPLEX parameters configuration. While investigating the impact of tuned software parameters on the performance, some contributions evidenced that improved parameter configurations may lead to substantial speedup for solving many combinatorial problems (Baz et al., 2007; Hutter et al., 2009 or Fawcett and Hoos, 2016). Pilot runs of our model on test instances led to the configuration of the CPLEX MIP solver for FFT as displayed in Table 2 whereas default parameter values were used for FFF. Table 2 provides the rationale for choosing some specific parameters values. Apart from the branching priority on the variables, the chosen CPLEX parameters values reflect an aggressive setting in searching for good feasible solutions.

Parameter name	Value	Description/Motivation
MIP dynamic search switch	1	Traditional branch-and-cut, no Dynamic Search for the branching phase. The choice is motivated by the fact that the Dynamic Search internals are largely undisclosed, so we went for a more reproducible behaviour of the solver.
Feasibility pump switch	2	Focus on finding solutions with better objective values, instead of potentially worse feasible ones (for further details on the method, see Fischetti et al., 2005).
RINS heuristic frequency	3	Application every 3 nodes of the Relaxation Induced Neighbourhood Search heuristic, an expensive heuristic useful to find high quality integer solutions (this heuristic is presented in Danna et al., 2005).
MIP probing level	2	Enforcement of an aggressive probing on variables before the branching phase.
MIP priority order switch	1	Required for the aforementioned branching priority rule, otherwise it would be ignored.
MIP repeat presolve switch	3	Repetition of the presolve to allow new cuts and new root cuts.
MIP dive strategy	3	The MIP traversal strategy occasionally performs probing dives, where it looks ahead before deciding which node to choose. With a value of 3, the solver is enabled to spend more time exploring potential solutions that are similar to the current one.

Table 2: CPLEX configuration for FFT

3.3 Results

3.3.1 Easy instances and small instances up to 50 jobs

Table 3 presents the number of optimal solutions (# Opt) and the associated average CPU time in seconds for each exact model on the instances of Elidrissi et al. (2021). Results are aggregated over the number of jobs and machines. The four exact approaches FFT, FFF, $\mathcal{F}2$ and TIV I were able to find the optimal solutions to all problems with $n \leq 50$ but with a similar and quite fast execution time only for $n = 10$ jobs. For $n = 50$, FFT and FFF strikingly outperformed $\mathcal{F}2$ and TIV I. For the most difficult problem $(n, m) = (50, 3)$, FFF was 5 times faster on average than $\mathcal{F}2$; and FFT, 35 times quicker. For $n = 100$, only FFF and FFT were able to find all optimal solutions, again with a quite small execution time compared with $\mathcal{F}2$. TIV I could find only 4 optimal solutions over 10, with the highest average CPU time. Due to its poor performance for $n = 100$, TIV I was run on Silva's instances up to $n = 50$ jobs.

			FFT		FFF		$\mathcal{F}2$		TIV I	
n	m	# Inst.	# Opt	CPU	# Opt	CPU	# Opt	CPU	# Opt	CPU
10	3	60	60	0.29	60	0.30	60	0.31	60	0.32
	4	60	60	0.10	60	0.15	60	0.14	60	0.14
20	3	60	60	2.17	60	14.96	60	11.50	60	23.13
50	3	20	20	16.55	20	115.34	20	586.54	20	1541.61
	7	20	20	2.69	20	20.91	20	62.05	20	164.01
100	5	10	10	1.41	10	3.86	9	542.33	4	1329.74

Table 3: Results on Elidrissi's instances

Table 4 displays the results of the 4 models over Silva's instances with up to $n = 50$ jobs. The number of optimal solutions and the corresponding average CPU time are provided for each combination of parameters values (n, m, α). The row labelled 'Overall' aggregates the performance indicators for each combination of jobs and machines (n, m). All exact approaches were able to find the optimal solutions for all instances with $n = 10, 20$ jobs. Once again, FFT and FFF were the fastest approaches, while TIV I was the slowest. For $n = 30$, only TIV I failed to find 2 optimal solutions over 60 with an average execution time that is more than twice that of $\mathcal{F}2$. FFT is 8 times faster than $\mathcal{F}2$ and FFF, more than 3 times quicker than $\mathcal{F}2$. Under $n = 50$, FFT found all optimal solutions, closely followed by FFF that missed only 3 of them over 60. The differences in the execution times are once again remarkable, with FFT and FFF being respectively 5 times and 3 times faster than $\mathcal{F}2$. With $n = 50$ jobs, $\mathcal{F}2$ found 53 optimal solution over 60 while TIV I reached 38 of them.

				FFT		FFF		$\mathcal{F}2$		TIV I	
n	m	α	# Inst.	# Opt	CPU	# Opt	CPU	# Opt	CPU	# Opt	CPU
10	2	0.1	5	5	0.83	5	0.67	5	0.68	5	0.57
		0.3	5	5	1.17	5	1.12	5	1.01	5	0.98
		0.5	5	5	1.58	5	1.57	5	1.69	5	1.42
	4	0.1	5	5	0.08	5	0.10	5	0.12	5	0.12
		0.3	5	5	0.17	5	0.22	5	0.18	5	0.24
		0.5	5	5	0.17	5	0.28	5	0.23	5	0.34
	6	0.1	5	5	0.04	5	0.07	5	0.07	5	0.07
		0.3	5	5	0.08	5	0.09	5	0.09	5	0.11
		0.5	5	5	0.07	5	0.09	5	0.09	5	0.09
Overall			45	45	0.47	45	0.47	45	0.46	45	0.44
20	2	0.1	5	5	2.77	5	4.41	5	8.27	5	27.75
		0.3	5	5	6.63	5	11.06	5	23.57	5	52.51
		0.5	5	5	8.60	5	22.82	5	34.01	5	118.78
	4	0.1	5	5	0.68	5	0.89	5	1.40	5	5.03
		0.3	5	5	1.60	5	4.27	5	7.93	5	9.82
		0.5	5	5	3.22	5	6.22	5	10.06	5	18.45
	6	0.1	5	5	0.30	5	0.62	5	1.22	5	1.77
		0.3	5	5	2.15	5	1.81	5	2.02	5	6.99
		0.5	5	5	2.35	5	3.59	5	6.19	5	6.85
	10	0.1	5	5	0.07	5	0.18	5	0.30	5	0.32
		0.3	5	5	0.13	5	0.31	5	0.50	5	0.47
		0.5	5	5	0.18	5	0.43	5	0.52	5	0.40
Overall			60	60	2.39	60	4.72	60	8	60	20.76
30	2	0.1	5	5	11.66	5	21.19	5	112.82	5	413.46
		0.3	5	5	63.49	5	95.22	5	255.99	5	740.92
		0.5	5	5	115.32	5	230.28	5	852.92	3	2761.62
	4	0.1	5	5	1.39	5	2.84	5	8.93	5	31.75
		0.3	5	5	7.36	5	39.78	5	121.08	5	384.45
		0.5	5	5	15.13	5	95.70	5	266.73	5	758.49
	6	0.1	5	5	1.08	5	2.48	5	6.29	5	22.80
		0.3	5	5	3.09	5	12.89	5	35.32	5	56.07
		0.5	5	5	13.06	5	32.44	5	278.61	5	361.84
	10	0.1	5	5	0.32	5	0.67	5	1.95	5	2.36
		0.3	5	5	1.25	5	2.07	5	2.90	5	3.61
		0.5	5	5	2.57	5	4.13	5	8.19	5	15.77
Overall			60	60	19.64	60	44.97	60	162.64	58	383.49
50	2	0.1	5	5	120.75	5	173.44	5	1150.54	2	1008.90
		0.3	5	5	416.77	5	1475.02	3	1650.70	0	-
		0.5	5	5	617.47	2	2281.86	0	-	0	-
	4	0.1	5	5	4.86	5	13.78	5	166.86	5	924.55
		0.3	5	5	47.02	5	449.22	5	1494.31	1	3488.81
		0.5	5	5	89.00	5	620.66	5	1629.86	0	-
	6	0.1	5	5	5.88	5	9.07	5	149.83	5	608.51
		0.3	5	5	13.63	5	75.94	5	255.49	5	769.08
		0.5	5	5	28.94	5	188.5	5	440.22	5	1075.99
	10	0.1	5	5	2.42	5	4.53	5	70.08	5	187.11
		0.3	5	5	5.44	5	22.41	5	78.25	5	192.31
		0.5	5	5	9.43	5	53.31	5	233.85	5	328.47
Overall			60	60	113.47	57	350.76	53	628.27	38	682.54

Table 4: Results on Silva's instances, $n \leq 50$

3.3.2 Medium-sized instances

Table 5 displays the results of FFT, FFF and $\mathcal{F}2$ on Silva's instances for $n = 75, 100$. FFT provided all optimal solutions but 2 out of 60 instances with 75 jobs, whereas $\mathcal{F}2$ was able to find less than half of them, with an average execution time about 3 times higher. With $n = 100$, the performance of $\mathcal{F}2$ dramatically decreased, reaching optimality in only 6 cases out of 60, compared to 51 optimal solutions for FFT and 44 for FFF.

				FFT		FFF		$\mathcal{F}2$	
n	m	α	# Inst.	# Opt	CPU	# Opt	CPU	# Opt	CPU
75	2	0.1	5	5	455.65	5	1459.59	0	-
		0.3	5	5	1770.33	0	-	0	-
		0.5	5	3	2048.22	0	-	0	-
	4	0.1	5	5	18.19	5	47.20	3	884.84
		0.3	5	5	232.87	4	2050.93	0	-
		0.5	5	5	604.42	1	3163.65	0	-
	6	0.1	5	5	15.05	5	40.57	3	2115.90
		0.3	5	5	86.67	5	347.14	4	2566.85
		0.5	5	5	203.37	5	1013.85	1	2468.04
	10	0.1	5	5	10.59	5	12.69	5	267.98
		0.3	5	5	17.90	5	78.05	5	564.06
		0.5	5	5	97.44	4	527.25	2	713.04
Overall			60	58	408.74	44	647.09	23	1188.00
100	2	0.1	5	5	1139.85	5	2406.22	0	-
		0.3	5	1	3477.94	0	-	0	-
		0.5	5	0	-	0	-	0	-
	4	0.1	5	5	33.64	5	140.56	1	1307.86
		0.3	5	5	1220.42	0	-	0	-
		0.5	5	5	2530.57	0	-	0	-
	6	0.1	5	5	33.17	5	88.66	1	1752.52
		0.3	5	5	290.13	5	1364.40	0	-
		0.5	5	5	811.19	1	2581.00	0	-
	10	0.1	5	5	6.40	5	26.35	3	2180.30
		0.3	5	5	43.73	5	215.93	1	3548.61
		0.5	5	5	210.33	5	1350.69	0	-
Overall			60	51	687.75	36	848.47	6	2191.65

Table 5: Results on Silva's instances, $n = 75, 100$

Our results in Tables 4 and 5 show that the performance of the models improve with the number of machines, as pointed out by Silva et al. (2023). With more machines, the horizon T decreases which implies a reduction in the number of variables. To illustrate, Figure 2 shows the evolution with the number m of machines, of the number of variables for the case $(n, m, \alpha) = (50, m, 0.1)$, for which FFF, FFT and $\mathcal{F}2$ reached the optimal solutions over the 5 draws of each combination. Obviously FFF and FFT have the same number of variables, which are notably lower than that of $\mathcal{F}2$. The second graph in Figure 2 is a box plot of the execution times in seconds across the draws for each value of m. It clearly illustrates the performance improvement as m increases since with it, there is a decrease in the number of variables. It also evidences the much lower execution times of FFF and FFT compared to $\mathcal{F}2$. The last graph in Figure 2 is a zoom of the CPU times of FFF and FFT and illustrates the superiority of FFT over FFF.

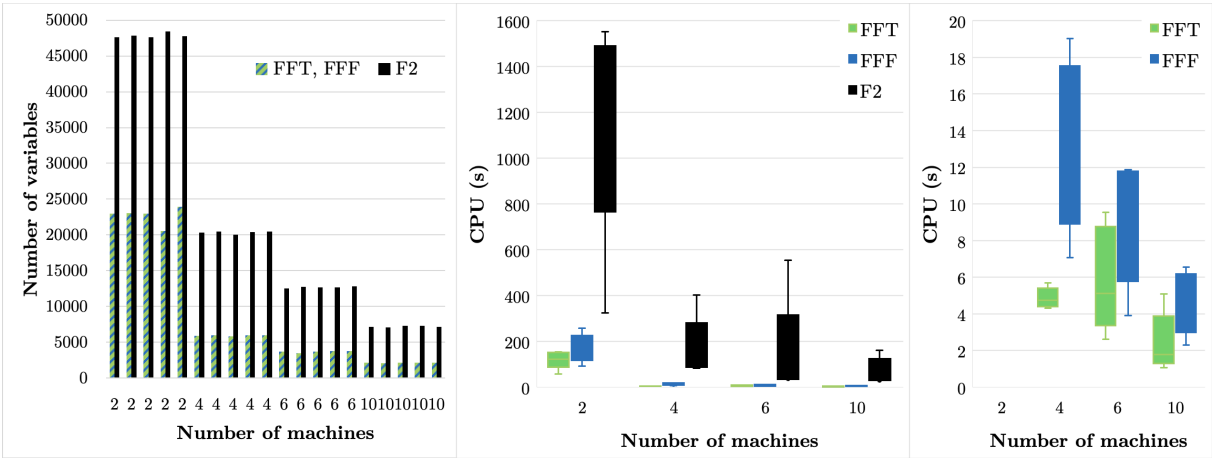

Figure 2: Evolution with the number of machines of the variables and execution times of FFT, FFF, $\mathcal{F}2$ for $n = 50$ and $\alpha = 0.1$

The good performance of the Flow-Flow formulations can also be explained by the high quality of their lower bounds provided by their continuous relaxation. The average deviation of optimal solutions from these lower bounds is 0.73% and 2.59% for FFT and FFF, respectively, across all 345 instances of Silva et al. (2023). The deviation is even zero for FFT in 170 instances out of 345, indicating that the continuous relaxation already provides the optimal value of the makespan. In contrast, the average deviation for $\mathcal{F}2$ amounts to 64.31% across all instances. Figure 3 plots the values of the continuous relaxation of FFF and $\mathcal{F}2$ as well as the optimal makespan, for the instances with $(n, m, \alpha) = (50, m, 0.5)$. As can be seen in Figure 3, the continuous relaxation of FFF remains quite close to the optimal makespan whereas that of $\mathcal{F}2$ stays far below, with improvements however as m increases. For $\mathcal{F}2$, the average deviation over the 90 instances with $m = 2$ is equal to 79.35% but drops to 54.81% on average over the 75 instances with $m = 10$ machines. It seems that both $\mathcal{F}2$ and TIV I suffer from the same min-max formulation, leading to poor lower bounds from their continuous relaxation. Over the 225 problems with $n \leq 50$ on which we also run TIV I, we got respectively an average deviation of optimal solutions to the lower bounds equal to 60.24% for TIV I and 62.28% for $\mathcal{F}2$. However for the same problems, FFF and FFT still produced extremely tight bounds, with an average deviation of 3.24% and 0.94% respectively.

Referring to the quality of lower bounds obtained from the continuous relaxation, it is worth mentioning that over all the 345 instances, all bounds of FFF were greater than $\mathtt{LB}_{\mathrm{PMTN}}$ versus only 11 of them from $\mathcal{F}2$. The bounds from the continuous relaxation of $\mathcal{F}2$ were 36.3% lower than $\mathtt{LB}_{\mathrm{PMTN}}$ on average while on the contrary those of FFF were 4% greater than $\mathtt{LB}_{\mathrm{PMTN}}$. This result substantiates the proof provided in Section 2.3.

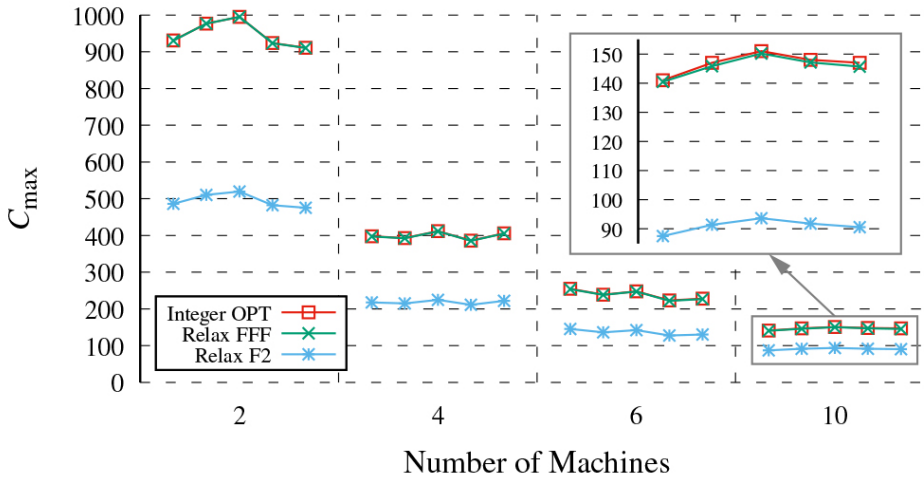

Figure 3: Optimal makespan and continuous relaxation of FFF and $\mathcal{F}2$ for $n = 50$ and $\alpha = 0.5$

3.3.3 Large instances with 150 and 200 jobs

Table 6 displays the results for the instances with $n = 150, 200$ jobs, applying again the 3 exact approaches. FFT and FFF were respectively able to optimally solve 60 and 43 problems out of a total of 90, whereas $\mathcal{F}2$ only reached 5 optimal solutions. These 5 problems actually correspond to 5 draws of a single combination of parameters ($n = 150$, $m = 10$, $\alpha = 0.1$) for which the average execution time of $\mathcal{F}2$ is about 90 times higher than that of FFF, and 290 times higher than that of FFT. As displayed in Table 6, the Flow-Flow formulations have an execution time of about 10 minutes on average.

				FFT		FFF		$\mathcal{F}2$	
n	m	α	# Inst.	# Opt	CPU	# Opt	CPU	# Opt	CPU
150	4	0.1	5	5	62.72	5	249.69	0	-
		0.3	5	0	-	0	-	0	-
		0.5	5	0	-	0	-	0	-
	6	0.1	5	5	33.47	5	145	0	-
		0.3	5	5	1424.84	0	-	0	-
		0.5	5	2	2299.4	0	-	0	-
	10	0.1	5	5	3.19	5	10.13	5	935.88
		0.3	5	5	216.33	5	743.14	0	-
		0.5	5	5	425.16	5	2107.66	0	-
Overall			45	32	482.11	25	651.12	5	935.88
200	4	0.1	5	5	208.62	5	569.59	0	-
		0.3	5	0	-	0	-	0	-
		0.5	5	0	-	0	-	0	-
	6	0.1	5	5	76.8	5	395.67	0	-
		0.3	5	3	1852.73	0	-	0	-
		0.5	5	0	-	0	-	0	-
	10	0.1	5	5	7.5	5	19.28	0	-
		0.3	5	5	524.66	3	2482.63	0	-
		0.5	5	5	1041.67	0	-	0	-
Overall			45	28	530.52	18	687.26	0	-

Table 6: Results on large-sized instances, $n = 150, 200$

Approximation results. Considering the large-sized instances with $n = 150, 200$ jobs, not all of them can be solved to optimality by exact approaches. Nevertheless, a relative optimality gap can be computed, using the best absolute gap reported by the solver when it reaches the time limit and the best known integer solution found by all known methods.

				Opt Gap (%)		
n	m	α	# Inst.	FFT	FFF	$\mathcal{F}2$
150	4	0.1	5	0.00	0.00	49.04
		0.3	5	1.61	1.63	49.22
		0.5	5	3.29	3.35	50.12
	6	0.1	5	0.00	0.00	39.04
		0.3	5	0.00	2.39	40.17
		0.5	5	2.26	5.68	45.16
	10	0.1	5	0.00	0.00	0.00
		0.3	5	0.00	0.00	37.77
		0.5	5	0.00	0.00	36.14
Overall			45	0.80	1.45	38.52
200	4	0.1	5	0.00	0.00	49.22
		0.3	5	1.46	1.50	49.55
		0.5	5	3.19	3.22	50.45
	6	0.1	5	0.00	0.00	49.02
		0.3	5	0.21	1.78	49.84
		0.5	5	5.08	5.15	50.81
	10	0.1	5	0.00	0.00	40.84
		0.3	5	0.00	0.35	50.88
		0.5	5	0.00	3.98	45.27
Overall			45	1.10	1.77	48.43

Table 7: Relative optimality gaps on large-sized instances, $n = 150, 200$

Table 7 reports the average relative optimality gaps for the three models FFT, FFF and $\mathcal{F}2$ over large-sized instances. A value of 0.00% in a model column means that all the 5 instances drawn from that combination of parameters (n, m, α) can be solved to optimality by the relevant model. It can be seen that relative optimality gaps obtained by FFT and FFF are significantly tighter than the ones provided by $\mathcal{F}2$. Motivation for this behaviour is twofold: on one side, FFT and FFF find good integer solutions faster than $\mathcal{F}2$; on the other side, the continuous relaxations of FFT and FFF are extremely tight (see Section 3.3.2 and Figure 3) providing better lower bounds and, consequently, better gaps than $\mathcal{F}2$.

4 Conclusion

In this paper, we have proposed an arc-flow formulation to solve the scheduling problem of a set of jobs that must be processed non preemptively on identical parallel machines and requiring prior setup operations on a common server, with the aim of minimising the makespan. Our model relies on an arc-flow formulation for both machines and server with shared variables related to the start time of the jobs and uses an efficient procedure for handling identical jobs. The resulting Flow-Flow formulations (FFF, FFT) therefore works with a fairly limited number of variables. Furthermore, the continuous relaxation of the Flow-Flow formulations produces high quality lower bounds that expedite the convergence towards optimal solutions. By contrast, the best formulation in past research, $\mathcal{F}2$, quite often generates lower bounds below a trivial hence poor lower bound to the problem. Computational experiments showed that starting from $n = 75$ jobs, the performance of $\mathcal{F}2$ clearly degraded, with 29 optimal solution out of the 120 instances from Silva et al. (2023) with $n = 75, 100$, compared to 109 optimal solutions for FFT (and 80 for FFF), and a 3 times faster execution time for FFT. Extended experiments with $n = 150, 200$ jobs again highlighted the good performance of our models as FFT and FFF were respectively able to optimally solve 60 and 43 problems out of 90 whereas $\mathcal{F}2$ only solved 5 instances to optimality. Our experimental framework revealed difficult instances to solve ($n > 100$ and $\alpha = 0.5$) suggesting that additional effort could be spent to enhance our solution method, with column generation as a promising avenue.

Besides, our Flow-Flow formulation could be adapted to several variants of the problem that appear in real settings such as configurations with two identical servers or a single server performing pre and post operations.

Appendix

In this appendix we illustrate the computations occurring in our implementation of the HS1 and HS2 heuristics by Elidrissi et al. (2018a). They are used in order to compute the time horizon

$$T = \min\{C_{\max}^{\text{HS1}}, C_{\max}^{\text{HS2}}\}.$$

We also illustrate the computation of the trivial preemptive lower bound $\texttt{LB}_{\text{PMNT}}$ and the improved $\texttt{LB}_{\text{Improved}}$.

We use an example with $n = 10$ jobs, $m = 3$ machines and job data as in Table 8, whose optimal value is known to be 103.

j	1	2	3	4	5	6	7	8	9	10	$\sum_{j=1}^{10} p_j = 245$
p_j	34	12	33	12	20	26	23	31	21	33	
s_j	6	3	5	5	5	6	8	7	2	8	$\sum_{j=1}^{10} s_j = 55$

Table 8: 10-job example

Both heuristics consider a number of job orderings, according to different priority rules summarised in Table 9. The dispatching heuristic (HS1/HS2) is run once for each priority rule and the best result is picked as proposed solution. We illustrate the workings of HS1/HS2 using only LPT, for the sake of simplicity. For a detailed pseudocode of the two heuristics we refer to Elidrissi et al. (2018a). Here we illustrate the logic behind the algorithms.

Ordering	Meaning	Break ties by
SPT	Shortest Processing Time: ascending order on p_j	SST
LPT	Longest Processing Time: descending order on p_j	LST
SST	Shortest Setup Time: ascending order on s_j	SPT
LST	Longest Processing Time: descending order of s_j	LPT
SCT	Shortest Completion Time: ascending order of $s_j + p_j$	SPT
LCT	Longest Completion Time: descending order of $s_j + p_j$	LPT

Table 9: Priority rules

HS1

HS1 builds a solution in two steps, aimed at minimising the idle time on the working machines. Let L be the LPT-ordered set of jobs. First of all the first $m-1$ jobs with the smallest setup times are extracted from L and scheduled on machines $1, \ldots, m-1$. From now on, the next job j to be scheduled is selected according to the following criteria.

- Job j will be scheduled on the earliest available machine k; let $C(M_k)$ be the time at which such a machine becomes available.
- Job j cannot start before the server has completed the last setup; letting $C(S)$ be the time at which the server becomes available, job j cannot start before time $t_a = \max\{C(S), C(M_k)\}$.
- The most favourable situation arises when the setup for job j occurs between time t_a and the completion time $C(M_{k'})$ of the second-earliest available machine k'

The job j to be scheduled is selected — if possible — as the first available job in L satisfying

$$s_j \leq C(M_{k'}) - t_a.$$

If no such job exists in L, the first job in L is anyway selected as the next job j.

The example in Table 8 is solved as follows. The $m - 1 = 2$ jobs with the smallest setup times are jobs $j = 9, 2$ that are scheduled on machines M_1, M_2. The LPT sequence for the remaining jobs is

$$L = (1, 10, 3, 8, 6, 7, 5, 4).$$

Iteration 1. The earliest available machine is M_3 at time $C(M_3) = 0$, but the server will not be available until time $C(S) = 5$, hence $t_a = 5$. The second-earliest available machine is M_2 with $C(M_2) = 17$; thus job 1 is selected, since $s_1 = 6 \leq C(M_2) - t_a$, and scheduled on M_3 at time 5.

Iteration 2. The earliest available machine is M_2 at time $C(M_2) = 17$. The server is already available, since it has completed the last setup at $C(S) = 11$. Hence $t_a = 17$. The second-earliest available machine is M_1 at $C(M_1) = 23$; thus job 3 is selected, since $s_3 = 5 \leq C(M_1) - t_a$. Note that job 10 is not selected at this step because $s_{10} = 8$, violating the latter condition.

Along the same lines, with the same criteria, jobs 10, 8, 6, 7 are scheduled in iterations 3, 4, 5, and 6.

Iteration 7 The earliest available machine is M_3 with $C(M_3) = 83$, and the server is already available (since time $t = 72$). Hence $t_a = 83$. The second earliest available machine is M_2 at $C(M_2) = 87$. No job among those remaining in L (specifically jobs $5, 4$) has a setup time satisfying $s_j \leq C(M_2) - t_a$. Hence job 5 is scheduled next on M_3 at $t = 83$.

The last job (job 4) is finally scheduled on machine M_2. The overall makespan is 108. The reader is referred to the Gantt chart in Figure 4.

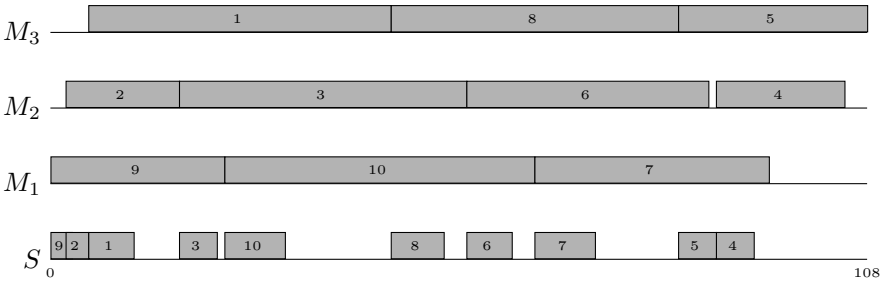

Figure 4: Schedule computed by HS1

HS2

HS2 builds a solution in three steps, aimed at minimising the idle time on the common server. Let L be the LPT-ordered set of jobs. First of all, the job with minimum p_j is taken and set aside, to be scheduled last. Then the first $m-1$ jobs in L are extracted and scheduled on machines $1, \ldots, m-1$. From now on, the next job j to be scheduled is selected accordingly with the following criteria.

- Job j will be scheduled on the earliest available machine k; let $C(M_k)$ be the time at which such machine becomes available.
- Job j cannot start before the server has completed the last setup; let $C(S)$ be the time at which the server becomes available, so j cannot start before $t_a = \max\{C(S), C(M_k)\}$.
- The most favourable situation arises when the setup for job j is longer than the interval between time t_a and the completion time $C(M_{k'})$ of machine k' which is the the second-earliest available machine

The job j to be scheduled is selected — if possible — as the first available job in L satisfying

$$s_j \geq C(M_{k'}) - t_a.$$

If no such job exists in L, the first job in L is anyway selected as the next job j.

The example in Table 8 is solved as follows. Job 2 with minimum processing time is set aside for being scheduled last. The LPT sequence for the remaining jobs is

$$L = (1, 10, 3, 8, 6, 7, 9, 5, 4).$$

The first two jobs in L (specifically 1 and 10) are scheduled on machines M_1, M_2.

Iteration 1. The earliest available machine is M_3 at time $C(M_3) = 0$, but the server will not be available until time $C(S) = 14$, hence $t_a = 14$. The second-earliest available machine is M_1 with $C(M_1) = 40$; no job with $s_j \geq C(M_1) - t_a$ is available in L, so job 3 is selected, being the next unscheduled job in L. It is scheduled at time $t = 14$ on M_3.

Iteration 2 The earliest available machine is M_1 at time $C(M_1) = 40$; the server is already available since time $C(S) = 19$, hence $t_a = 40$. The second-earliest available machine is M_2 with $C(M_2) = 47$. Now job 8 from L satisfies $s_8 = 7 \geq C(M_2) - t_a$. Thus job 8 is scheduled on M_1 at time $t = 40$.

With similar criteria, jobs $6, 7, 9, 5, 4$ are selected in the successive iterations. Finally, job 2 is scheduled on M_1 at $t = 101$. The resulting makespan is 116. The reader is referred to the Gantt chart in Figure 5.

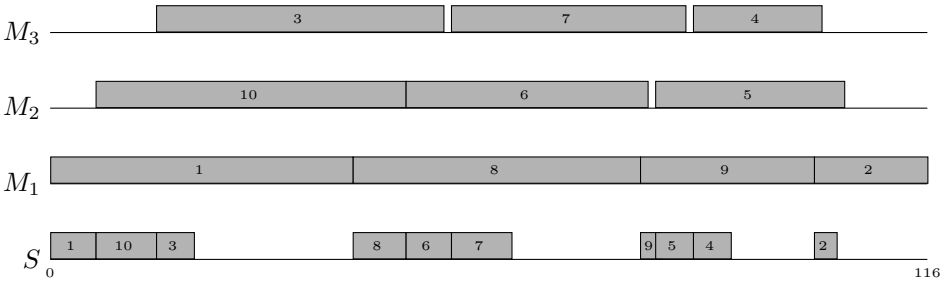

Figure 5: Schedule computed by HS2

Lower bounds

The preemptive lower bound is computed by equation (2):

$$\mathtt{LB}_{\mathrm{PMTN}} = \frac{1}{m}\left[\sum_{j=1}^{n} s_j + \sum_{j=1}^{n} p_j\right] = \frac{1}{3}(55 + 245) = 100.$$

For $\mathtt{LB}_{\mathrm{Improved}}$ we use the SST sequence

$$\sigma = (9, 2, 4, 5, 3, 6, 1, 8, 7, 10),$$

and compute the two components

$$\sum_j s_j + \min_j p_j = 55 + 12 = 67$$

$$\mathtt{LB}_{\mathrm{PMTN}} + \frac{\sum_{j=1}^{m-1}(m-j)s_{\sigma(j)}}{m} = 100 + \frac{2 \cdot 2 + 1 \cdot 3}{3} = 100 + \frac{7}{3} \approx 102.33.$$

The first component of the bound takes into account an infinite-capacity relaxation of the working machines that allows continuous operations on the server; the second component takes into account a preemptive relaxation on the working machines plus a contribution from the minimum possible idle time required in the first operation on each machine. Hence the value for the bound is

$$\mathtt{LB}_{\mathrm{Improved}} = \max\{67, 102.33\} = 102.33.$$

Note that in this example, the continuous relaxation of $\mathcal{F}2$ gives a value of 69.93, the relaxation of FFF gives a value of 102.28, which is much stronger than $\mathtt{LB}_{\mathrm{PMTN}}$ and $\mathcal{F}2$, practically comparable to $\mathtt{LB}_{\mathrm{Improved}}$.

Comparing the performance of ATIF with $\mathcal{F}2$ and FFT

Table 10 provides a comparison of our model FFT with $\mathcal{F}2$ and ATIF on the same instances as those in Silva et al. (2023), up to 30 jobs, since ATIF was unable to provide any solutions for instances with more than 20 jobs. Moreover, ATIF struggled to solve most of the problems with 20 jobs, and for the few problems it could solve, the CPU time was significantly higher than that of FFT.

				ATIF		$\mathcal{F}2$		FFT	
n	m	α	# Inst.	# Opt.	Time (s)	# Opt.	Time (s)	# Opt.	Time (s)
10	2	0.1	5	5	26.90	5	0.68	5	0.83
		0.3	5	5	60.34	5	1.01	5	1.17
		0.5	5	5	70.35	5	1.69	5	1.58
	4	0.1	5	5	2.98	5	0.12	5	0.08
		0.3	5	5	3.55	5	0.18	5	0.17
		0.5	5	5	4.62	5	0.23	5	0.17
	6	0.1	5	5	0.96	5	0.07	5	0.04
		0.3	5	5	0.67	5	0.09	5	0.08
		0.5	5	5	0.45	5	0.09	5	0.07
Overall			45	45	18.98	45	0.46	45	0.47
20	2	0.1	5	1	3176.43	5	8.27	5	2.77
		0.3	5	0	-	5	23.57	5	6.63
		0.5	5	0	-	5	34.01	5	8.60
	4	0.1	5	0	-	5	1.40	5	0.68
		0.3	5	0	-	5	7.93	5	1.60
		0.5	5	0	-	5	10.06	5	3.22
	6	0.1	5	1	3494.54	5	1.22	5	0.30
		0.3	5	0	-	5	2.02	5	2.15
		0.5	5	0	-	5	6.19	5	2.35
	10	0.1	5	5	86.21	5	0.30	5	0.07
		0.3	5	5	254.71	5	0.50	5	0.13
		0.5	5	5	157.46	5	0.52	5	0.18
Overall			60	17	2697.44	60	8.00	60	2.39
30	2	0.1	5	0	-	5	112.82	5	11.66
		0.3	5	0	-	5	255.99	5	63.49
		0.5	5	0	-	5	852.92	5	115.32
	4	0.1	5	0	-	5	8.93	5	1.39
		0.3	5	0	-	5	121.08	5	7.36
		0.5	5	0	-	5	266.73	5	15.13
	6	0.1	5	0	-	5	6.29	5	1.08
		0.3	5	0	-	5	35.32	5	3.09
		0.5	5	0	-	5	278.61	5	13.06
	10	0.1	5	0	-	5	1.95	5	0.32
		0.3	5	0	-	5	2.90	5	1.25
		0.5	5	0	-	5	8.19	5	2.57
Overall			60	0	-	60	162.64	60	19.64

Table 10: Performance of ATIF, $\mathcal{F}2$, FFT on instances with up to 30 jobs

References

Abdekhodaee, A.H., Wirth, A., 2002. Scheduling parallel machines with a single server: some solvable cases and heuristics. Computers and Operations Research 29, 295–315.

Abu-Shams, M., Ramadan, S., Al-Dahidi, S., Abdallah, A., 2022. Scheduling large-size identical parallel machines with single server using a novel heuristic-guided genetic algorithm (das/ga) approach. Processes 10, 1–18.

Baz, M., Hunsaker, B., Brooks, J., Gosavi, A., 2007. Automated tuning of optimization software parameters. Technical Report 7. University of Pittsburgh Department of Industrial Engineering.

Bektur, G., Saraç, T., 2019. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. Computers and Operations Research 103, 46–63.

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

Cheng, T., Kravchenko, S.A., Lin, B.M., 2017. Preemptive parallel-machine scheduling with a common server to minimize makespan. Naval Research Logistics 64, 388–398.

Danna, E., Rothberg, E., Le Pape, C., 2005. Exploring relaxation induced neighborhoods to improve mip solutions. Mathematical Programming 102, 71–90.

Elidrissi, A., Benbrahim, M., Benmansour, R., Duvivier, D., 2018a. Greedy heuristics for identical parallel machine scheduling problem with single server to minimize the makespan, in: MATEC Web of Conferences, International Workshop on Transportation and Supply Chain Engineering (IWTSCE'18). pp. 1–7.

Elidrissi, A., Benbrahim, M., Benmansour, R., Duvivier, D., Sifaleras, A., Mladenovic, N., 2020. Variable neighborhood search for identical parallel machine scheduling problem with a single server, in: Variable Neighborhood Search, 7th International Conference on Variable Neighborhood Search (ICVNS 2019), Springer International Publishing. pp. 112–125.

Elidrissi, A., Benmansour, R., Benbrahim, M., Duvivier, D., 2018b. Mip formulations for identical parallel machine scheduling problem with single server, in: 4th International Conference on Optimization and Applications (ICOA), IEEE. pp. 1–6.

Elidrissi, A., Benmansour, R., Benbrahim, M., Duvivier, D., 2021. Mathematical formulations for the parallel machine scheduling problem with a single server. International Journal of Production Research 59, 6166–6184.

Elidrissi, A., Benmansour, R., Sifaleras, A., 2022. General variable neighborhood search for the parallel machine scheduling problem with two common servers. Optimization Letters in press. URL: `https://doi.org/10.1007/s11590-022-01925-2`.

Fawcett, C., Hoos, H.H., 2016. Analysing differences between algorithm configurations through ablation. Journal of Heuristics 22, 431–458.

Fischetti, M., Glover, F., Lodi, A., 2005. The feasibility pump. Mathematical Programming 104, 91–104.

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

Glass, C.A., Shafransky, Y.M., Strusevich, V.A., 2000. Scheduling for parallel dedicated machines with a single server. Naval Research Logistics 47, 304–328.

Graham, R., Lawler, E., Lenstra, J., Kan, A., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Hammer, P., Johnson, E., Korte, B. (Eds.), Discrete Optimization II. Elsevier. volume 5 of *Annals of Discrete Mathematics*, pp. 287–326.

Hall, N.G., Potts, C.N., Sriskandarajah, C., 2000. Parallel machine scheduling with a common server. Discrete Applied Mathematics 102, 223–243.

Hamzadayi, A., Yildiz, G., 2017. Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. Computers and Industrial Engineering 106, 287–298.

Hasani, K., Kravchenko, S.A., Werner, F., 2016. Minimizing the makespan for the two-machine scheduling problem with a single server: Two algorithms for very large instances. Engineering Optimization 48, 173–183.

Huang, S., Cai, L., Zhang, X., 2010. Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. Computers and Industrial Engineering 58, 165–174.

Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T., 2009. Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36, 267–306.

Kerkhove, L.P., Vanhoucke, M., 2014. Scheduling of unrelated parallel machines with limited server availability on multiple production locations: a case study in knitted fabrics. International Journal of Production Research 52, 2630–2653.

Kim, M.Y., Lee, Y.H., 2012. Mip models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Computers and Operations Research 39, 2457–2468.

Koulamas, C., 1996. Scheduling two parallel semiautomatic machines to minimize machine interference. Computers and Operations Research 23, 945–956.

Kramer, A., Dell'Amico, M., Feillet, D., Iori, M., 2020. Scheduling jobs with release dates on identical parallel machines by minimizing the total weighted completion time. Computers and Operations Research 123, 105018.

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

Kramer, A., Lalla-Ruiz, E., Iori, M., Voß, S., 2019b. Novel formulations and modeling enhancements for the dynamic berth allocation problem. European Journal of Operational Research 278, 170–185.

Kravchenko, S., Werner, F., 1997. Parallel machine scheduling problems with a single server. Mathematical and Computer Modelling 26, 1–11.

Kravchenko, S.A., Werner, F., 2001. A heuristic algorithm for minimizing mean flow time with unit setups. Information Processing Letters 79, 291–296.

Liu, G.S., Li, J.J., Yang, H.D., Huang, G.Q., 2019. Approximate and branch-and-bound algorithms for the parallel machine scheduling problem with a single server. Journal of the Operational Research Society 70, 1554–1570.

Martinovic, J., Scheithauer, G., Valério de Carvalho, J., 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, 458–471.

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

Silva, J.M.P., Subramanian, A., Uchoa, E., 2023. On time-indexed formulations for the parallel machine scheduling problem with a common server. Engineering Optimization xx, 1–18.

Silva, J.M.P., Teixeira, E., 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, 444–457.