Greedy heuristics for identical parallel machine scheduling problem with single server to minimize the makespan

Abdelhak El Idrissi, Mohammed Benbrahim, Rachid Benmansour, David Duvivier

Greedy heuristics for identical parallel machine scheduling problem with single server to minimize the makespan

Abdelhak El idrissi[1,3,⋆], *Mohammed* Benbrahim[1], *Rachid* Benmansour[2], and *David* Duvivier[3]

[1] *Laboratory MASI, Research team MOAD6, Ecole Mohammadia d'Ingénieurs, Rabat, Morocco*
[2] *Laboratory SI2M, Institut National de Statistique et d'Economie Appliquée , Rabat, Morocco*
[3] *LAMIH UMR CNRS 8201, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France*

Abstract. This paper considers the problem of scheduling a set of n independent jobs on m identical parallel machines with setup times constraints. Immediately before processing, each job must be loaded on a common server to perform the setup operation. During the loading operation, both the machine and the server are occupied. Two greedy heuristics are developed for ($m \geq 2$) in order to minimize respectively the server waiting time and the machine idle time. These heuristics generalize those proposed in the literature for the case of two machines. The computational results show the efficiency of the proposed heuristics.

1 Introduction

In this paper, we consider a parallel machine scheduling problem with a single server (PSS) and the minimization of the makespan. Throughout this paper we consider the static version of the problem were all jobs are available at the beginning of the schedule. Scheduling problem under server constraints are intensively studied, only for special cases such as equal setup time, equal processing time and equal length (the sum of setup time and processing time), regular job constraints or the case of two machines. In this paper, the case of an arbitrary number of machines for the PSS problem is studied.

The PSS problem is denoted by $Pm, S1|s_j, p_j|C_{max}$ using the standard scheduling notation, where m represents the number of identical parallel machines, $S1$ represents the only available server, and s_j and p_j are respectively the setup time of job j and its processing time. The objective function considered in this problem is the minimization of the makespan C_{max}. The complexity has been studied by Kravchenko and Werner [1]. The authors showed that the problem $Pm, S1|s_j = 1|C_{max}$ is unary NP-hard and analyze several list scheduling heuristics. The complexity for the case of two machines and also an arbitrary number of machines is discussed by Bruker *et al.* [2] for all of the well-known objective functions C_{max}; $\sum C_j$; $\sum w_j C_j$; $\sum U_j$, $\sum w_j U_j$; $\sum T_j$; $\sum w_j T_j$ and L_{max}.

In 2006 Abdelkhodaee *et al.* [5] extended their previous study in [4],[3] dealing with the regular case where $p_i - p_j \leq s_j \ \forall i, j$ and the case of equal setup time and equal processing (see also [13]) for the $P2, S1|s_j, p_j|C_{max}$. The authors proposed two greedy heuristics, a genetic algorithm, and the Gilmore-Gomory algorithm for the general case of the $P2, S1||C_{max}$. The analysis started in [5] has been extended by Gan *et al.* [6], in which two mixed integer linear programming formulations and two variants of a branch-and-price scheme were developed.

Moreover, Hasani *et al.* [7] developed two metaheuristics namely simulated annealing (SA) and genetic algorithm (GA), for the problem $P2, S1||C_{max}$. The results obtained are much better than all the previous algorithms and models proposed in [5], [6]. Hasani *et al.* [8] proposed also two algorithms to solve very large instances for $P2, S1||C_{max}$. The computational results showed that the two algorithms outperformed the precedent methods existing in the literature, however, for small-sized and medium-sized instances, the results obtained were not promising.

Recently, Arnaout [9] proposed an ant colony optimization (ACO) metaheuristic for $P2, S1||C_{max}$. The computational results showed that the proposed method outperformed the metaheuristics (SA) and (GA) proposed in [7] for large instances of the problem. The problem with several machines ($m \geq 3$) was only studied by Kim and Lee [10]. The authors proposed two mixed integer programming formulations. The first one is developed using assignment and positional date variables, and the second one is developed by adding the concept of the server waiting time. A hybrid heuristic algorithm combining simulated annealing and tabu search is also suggested.

The remainder of this paper is organized as follows. In Section 2, we present the problem formally. In Section 3, two greedy heuristics algorithms for an arbitrary number of machines are proposed with two lower bounds. We conduct computational experiments in Section 4 to show the performance of those heuristics in comparison with the literature. The conclusion and future works end this paper.

⋆e-mail: abdelhakelidrissi@research.emi.ac.ma

2 Problem description

The aim of this section is to give a detailed description of the PSS problem. It can be described as follows. Suppose there are m identical parallel machines which must process n independent jobs. Each job i has a known integer processing time p_i. Immediately before its processing, job i must be loaded on a machine by the server. This duration, which can be also considered as a setup operation has a known integer value s_i. During the setup operation, both the machine and the server are occupied and after loading a job on a particular machine the server becomes available for loading the next job. The processing operation starts immediately after the end of setup. The preemption during setup and processing operations of jobs is not allowed. The objective is to find a feasible schedule that minimizes the makespan.

The following notations are used to define the problem throughout the paper:

- n: number of jobs.
- m: number of machines.
- $N = \{1, \ldots, n\}$: set of jobs.
- $M = \{1, \ldots, m\}$: set of machines.
- p_i: the processing time of the job i.
- s_i: the setup time of the job i.
- $A = \{A_1, A_2, \ldots, A_n\}$: the list of jobs to be scheduled.
- $B = \{B_1, B_2, \ldots, B_n\}$: the scheduled list sequence obtained by the heuristic.
- $St(J_i)$: the starting time of the setup operation of job i.
- $C(J_i)$: the completion time of the processing operation of job i.
- $C(M_k)$: the completion time of the last scheduled job on machine k.
- R_1: Sort the jobs in increasing order of their processing times (SPT).
- R_2: Sort the jobs in decreasing order of their processing times (LPT).
- R_3: Sort the jobs in increasing order of their setup times (SST).
- R_4: Sort the jobs in decreasing order of their processing times (LST).
- R_5: Sort the jobs in increasing order of the sum of their processing and setup times (SPST).
- R_6: Sort the jobs in decreasing order of the sum of their processing and setup times (LPST).

3 Two Greedy heuristics

In this section, we propose two complementary greedy heuristics called HS1 and HS2 for the PSS problem under consideration in order to minimize respectively the server waiting time and the machine idle time. Indeed, it has been shown that the minimization respectively of the total server waiting time (i.e. the gaps between the loading of two jobs) and the minimization of the machine idle time is equivalent to the minimization of the makespan (see Kim and Lee [10]). The basic aim at each step of the two heuristics is, if possible, not to generate any machine idle time or any server waiting time.

3.1 Greedy heuristic HS1

This heuristic HS1 aims, to minimize machine idle time (the time machines are idle due to unavailability of the server). The pseudo-code of the proposed heuristic is as follows (see Algorithm 1).

In the first step jobs are arranged in a list in increasing order of their setup times. Then the first available $m-1$ jobs of the list are sequenced on the first available machines. After that, we sort the available unsequenced jobs according to some criterion and then choose an eligible job (i.e. one that would not create any machine idle time). If no eligible jobs exist we select the first job in the list. The list may be arranged in decreasing or increasing order of job lengths, setup times or processing times.

3.1.1 Illustrative example

Given an instance of $n = 10$ jobs and $m = 3$ machines. The processing times p_i and setup times of the jobs s_i are given in Table 1. The makespan value can be obtained by applying the heuristic HS1 as given above. It takes 0.002 seconds to solve the problem with the heuristics HS1. All the heuristics have been implemented using C++ language on a personal computer Intel(R) Core(TM) i5-2410M CPU @ 2.30 GHz with 8 GB of RAM memory. The feasible sequence is shown in Figure 1. With a value of the objective function (makespan) of 53.

Job	1	2	3	4	5	6	7	8	9	10
p_i	9	11	13	6	8	14	9	8	7	9
s_i	2	4	6	1	3	10	5	4	3	7

Table 1: Example instance with ($n = 10, m = 3$)

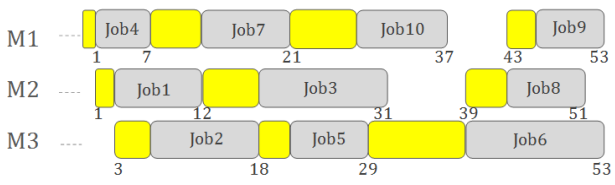

Figure 1: Feasible schedule for the ($n = 10, m = 3$) problem instance obtained by the heuristic HS1

In this illustrative example, we sort the initial list of jobs in increasing order of their setup times, then we schedule the first $m-1$ jobs on the first available machines, after this step, we sort for the second time the remaining jobs with the R_2 rule (longest processing time) and we start the job selection. Thus job $J4$ and job $J1$ are considered

Algorithm 1: HS1 Heuristic

Sort the initial list $A = \{A_1, A_2, \ldots, A_n\}$ in increasing order of setup times
$B \leftarrow \emptyset$
for $k = 1 \quad to \quad m-1$ **do**
  $B_k \leftarrow A_k, \quad A_k \leftarrow \emptyset$
  Execute B_k on Machine M_k
  $C(B_k) = St(B_k) + s(B_k) + p(B_k)$
  $St(B_{k+1}) = St(B_k) + s(B_k)$
  $C(M_k) = C(B_k)$
  $B \leftarrow B \cup \{B_k\}$
Sort the remaining list A according to one of the scheduling rule R_i
$i \leftarrow m, \quad t \leftarrow m$
$j \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
while $i \leq n-1$ **do**
  test=0
  for $k = m \quad to \quad n-1$ **do**
    if $(C(M_j) \geq \max\{St(B_{i-1}) + s(B_{i-1}); C(M_t)\} + s(A_k))$ **then**
      $test \leftarrow 1$
      $B_i \leftarrow A_k$
      Execute B_i on Machine M_t
      $C(B_i) = St(B_i) + s(B_i) + p(B_i)$
      $A \leftarrow A \setminus \{A_k\}, \quad B \leftarrow B \cup \{B_i\}$
      $C(M_t) = C(B_i)$
      $t \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
      $j \leftarrow \underset{v \in M \setminus M_t}{\text{argmin}}(C(M_v))$
      $St(B_{i+1}) = \max\{St(B_i) + s(B_i); C(M_t)\}$
      $i \leftarrow i + 1$
      break
  if *test=0* **then**
    for $k = m \quad to \quad n-1$ **do**
      if $(C(M_j) \leq \max\{St(B_{i-1}) + s(B_{i-1}); C(M_t)\} + s(A_k))$ **then**
        $test \leftarrow 1$
        $B_i \leftarrow A_k$
        Execute B_i on Machine M_t
        $C(B_i) = St(B_i) + s(B_i) + p(B_i)$
        $A \leftarrow A \setminus \{A_k\}, \quad B \leftarrow B \cup \{B_i\}$
        $C(M_t) = C(B_i)$
        $t \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
        $j \leftarrow \underset{v \in M \setminus M_t}{\text{argmin}}(C(M_v))$
        $St(B_{i+1}) = \max\{St(B_i) + s(B_i); C(M_t)\}$
        $i \leftarrow i + 1$
        break
Execute B_n on Machine M_t
$C_{max} \leftarrow \max\{C(M_1), ..., C(M_m)\}$

as the first jobs. Then, the third job must be scheduled on the first available machine (machine $M3$) and a decision must be made according to the end of the setup time of the last scheduled job $J1$, the completion time of the last job scheduled on the next available machine, in this step the machine $M1$ was the next available one and the completion time of last job scheduled on the current available machine. Indeed, to minimize the machine idle time as much as possible, the best way is to choose a job whose loading time is less than or equal to the completion time of the next available machine minus the maximum value of the end of setup time of the last scheduled job in the finale list B and the completion time of the last job scheduled on the current available machine. Thus, the job $J2$ has to be selected as the third job. The fourth unscheduled job, which has to be determined, must be scheduled on the first available machine (machine $M1$) and must have a set-up time less than or equal to the completion time of the next available machine (Machine $M2$) minus the maximum value of the end of setup time of the last scheduled job (job $J2$) and the completion time of the last job scheduled on the current available machine (job $J4$). Among the remaining unscheduled jobs, job $J7$ will be chosen. And so on until scheduling all of the remaining jobs.

3.2 Greedy Heustic HS2

In this section, we present the second greedy heuristic HS2 to minimize the gap between the completion time of the setup operation of a job and the start of setup operation of the next job. Contrarily to the heuristic Min-loagap proposed for the case of two machines by Hasani *et al.* [8] where the jobs were scheduled in staggered order on machines (i.e. jobs alternate between machines), in this heuristic the jobs are scheduled according to the availability of the machines. And the job with the minimal processing time is considered as the last job to be scheduled in the final sequence.

3.2.1 Illustrative example

Given an instance of $n = 10$ jobs and $m = 4$ machines. The processing times p_i and setup times of the jobs s_i are given in Table 2. The makespan value can be obtained by applying the heuristic HS2 as given above. It takes 0.002 seconds to solve the problem with the heuristics HS2. The feasible sequence is shown in Figure 2. With a value of the objective function (makespan) of 90. In this example equality $C_{max} = LB$ holds.

Job	1	2	3	4	5	6	7	8	9	10
p_i	2	4	8	3	5	6	7	9	7	2
s_i	8	7	12	4	8	9	10	14	11	5

Table 2: Example instance with $(n = 10, m = 4)$

In this illustrative example, we sort the initial list of jobs according to the longest processing time rule (R_2), then we schedule the first $m-1$ jobs on the first available machines, and the job with the minimal processing time

Algorithm 2: HS2 Heuristic

Sort the initial list $A = \{A_1, A_2, \ldots, A_n\}$ according to one of the scheduling rules R_i
$B \leftarrow \emptyset$
Find the job $A_k \in A$, $k \in \{1, .., n\}$ with the smallest processing time
$B_n \leftarrow A_k$
$A_t \leftarrow A_{t+1} \quad \forall t \in \{k, .., n-1\}$
for $k = 1 \quad to \quad m-1$ **do**
  $B_k \leftarrow A_k, \quad A_k \leftarrow \emptyset$
  Execute B_k on Machine M_k
  $C(B_k) = St(B_k) + s(B_k) + p(B_k)$
  $St(B_{k+1}) = St(B_k) + s(B_k)$
  $C(M_k) = C(B_k)$
  $B \leftarrow B \cup \{B_k\}$
$i \leftarrow m, \quad t \leftarrow m$
$j \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
while $i \leq n-1$ **do**
  test=0
  for $k = m \quad to \quad n-1$ **do**
    if $(C(M_j) \leq \max\{St(B_{i-1}) + s(B_{i-1}); C(M_t)\} + s(A_k))$ **then**
      $test \leftarrow 1$
      $B_i \leftarrow A_k$
      Execute B_i on Machine M_t
      $C(B_i) = St(B_i) + s(B_i) + p(B_i)$
      $A \leftarrow A \setminus \{A_k\}, B \leftarrow B \cup \{B_i\}$
      $C(M_t) = C(B_i)$
      $t \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
      $j \leftarrow \underset{v \in M \setminus M_t}{\text{argmin}}(C(M_v))$
      $St(B_{i+1}) = \max\{St(B_i) + s(B_i); C(M_t)\}$
      $i \leftarrow i + 1$
      break
  if *test=0* **then**
    for $k = m \quad to \quad n-1$ **do**
      if $(C(M_j) \geq \max\{St(B_{i-1}) + s(B_{i-1}); C(M_t)\} + s(A_k))$ **then**
        $test \leftarrow 1$
        $B_i \leftarrow A_k$
        Execute B_i on Machine M_t
        $C(B_i) = St(B_i) + s(B_i) + p(B_i)$
        $A \leftarrow A \setminus \{A_k\}, B \leftarrow B \cup \{B_i\}$
        $C(M_t) = C(B_i)$
        $t \leftarrow \underset{h \in M}{\text{argmin}}(C(M_h))$
        $j \leftarrow \underset{v \in M \setminus M_t}{\text{argmin}}(C(M_v))$
        $St(B_{i+1}) = \max\{St(B_i) + s(B_i); C(M_t)\}$
        $i \leftarrow i + 1$
        break
Execute B_n on Machine M_t
$C_{max} \leftarrow \max\{C(M_1), ..., C(M_m)\}$

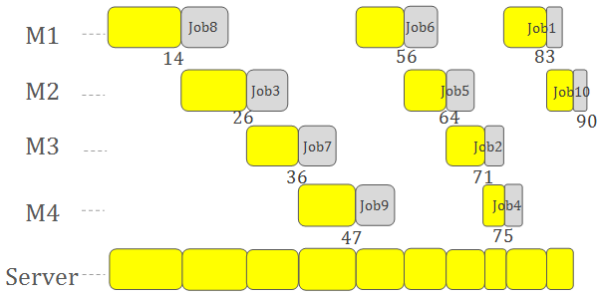

Figure 2: Optimal schedule for the $(n = 10, m = 4)$ problem instance obtained by the heuristic HS2

is scheduled in the last position of the schedule. Thus job $J8$, job $J3$ and job $J7$ are considered as the first jobs. After this step, we start the job selection. The fourth job must be scheduled on the first available machine and a decision must be made according to the end of the setup time of the last scheduled job $J7$, the completion time of the last job scheduled on the next available machine, in our case the machine $M1$ was the next available one and the completion time of last job scheduled on the current available machine. Indeed, to minimize the server waiting time as much as possible, the best way is to choose a job whose loading time is greater than or equal to the completion time of the next available machine minus the maximum value of the end of setup time of the last scheduled job in finale list B and the completion time of last job scheduled on the current available machine. Thus, the job $J9$ has to be selected as the fourth job. The fifth unscheduled job, which has to be determined, must be scheduled on the first available machine (machine $M1$) and must have a set-up time greater or equal than the completion time of the next available machine (machine $M2$) minus the maximum value of the end of setup time of the last scheduled job and the completion time of last scheduled on the current machine (job $J8$). Among the remaining unscheduled jobs, job $J6$ will be chosen. And so on until scheduling all of the remaining jobs.

3.3 Lower bounds

The investigation of lower bounds is useful for benchmarking heuristic solutions. Thus, to evaluate the quality of the solutions, the following lower bounds are proposed.

Proposition 1.

$$LB1 = \frac{1}{m} \sum_{1 \leq i \leq n} \{p_i + s_i\}$$

is a lower bound for the problem of $P, S||C_{max}$.

Proof. by contradiction:
if $LB1 \leq \frac{1}{m} \sum_{1 \leq i \leq n} \{p_i + s_i\}$

In contradiction with the fact that the makespan cannot take a value less than the sum of the processing times and the setup times of all jobs divided by the number of machines. □

Proposition 2.

$$LB2 = \sum_{1\leq i\leq n} s_i + \min_{1\leq i\leq n} p_i$$

is a a lower bound for the problem of $P, S\|C_{max}$.

Proof. In an optimal schedule, the solution will have a value of $\sum_{1\leq j\leq n} s_j$ plus the processing time of the last scheduled job. Indeed, the makespan is equal to the sum of setup times plus the total server waiting times as shown by Kim et al. [10] i.e. $C_{max} = \sum_{1\leq j\leq n} s_j + SWTT$ (with: $SWTT$ is the total server waiting time) and in an optimal schedule the total server waiting time will be equal to the processing time of the last executed job. So to obtain a good lower bound, the last scheduled job in the sequence should have the smallest processing time. □

Combining the two lower bounds $LB1$ and $LB2$, we obtain an overall lower bound $LB = \max\{LB1; LB2\}$ on the value of the makespan solution C_{max}. To show the effectiveness of the lower bound, we have compared it with the linear relaxation of the MIP models proposed by El idrissi *et al.* [11] for several instances. We have found that our LB is approximately equal to the linear relaxation for the majority of the cases, which shows its effectiveness.

4 Computational experiments

4.1 Description of data

In this section, we conduct a computational experiment where the above, heuristics HS1, HS2 and lower bounds are implemented. The data was generated in the same way as described firstly by Koulamas [12], and also described lately by Abdekhodaee and Wirth [3] and Hasani *et al.* [8]. The data are generated in the uncorrelated case, where the processing time values p_j are generated from a discrete uniform distribution $U(0, 100)$ and setup time values s_j are generated from a discrete uniform distribution $U(0, 100L)$ where $L = E(s_j)/E(p_j)$ is the server load and $E(x)$ denotes the mean of x.

HS1 was used for instances with $LB = LB1$, and HS2 for the instances with $LB = LB2$. From the experiments, it turned out that for all instances with $L \in \{0.1, 0.5, 0.8\}$, equality $LB = LB1$ always holds and for all instances with $L \in \{1.5, 1.8, 2\}$, equality $LB = LB2$ always holds. However, for the instances with $L = 1$, both lower bounds have to be taken into account. In the following, if $L \in \{0.1, 0.5, 0.8\}$ HS1 is only used and if $L \in \{1.5, 1.8, 2\}$ HS2 is only used.

In order to compare the performance of our proposed heuristics with the literature, three factors are used: a fixed number of machines, the problem size fixed to three levels: small-size instances with $n \in (8, 20)$, medium-size instances with $n \in (30, 40, 50, 100)$ and large-size instance with $n \in (200, 250, 300, 350)$. The server load is also fixed to $L \in \{0.1, 0.5, 0.8, 1.5, 1.8, 2\}$, for each value of the server load 5 instances are generated randomly for $n \in \{8, 20\}$ and 10 instances were generated randomly for each of the other value of n, m and L.

4.2 Performances

To evaluate the efficiency of the proposed heuristics, computational experiments were designed and conducted. The computational results with HS1 for small, medium and large-sizes instances are given in Table 3 and Table 4 and the computational results with HS2 for small, medium and large sizes instances are given in Table 5 and Table 6. In Tables 3, 4, 5, 6 , column 1 gives the number of jobs, column 2 gives the number of machines, column 4 until the last column give the values R_{avg}, denoting the average value of the ratio C_{max}/LB, and R_{max}, denoting the maximum value of the ratio C_{max}/LB among all instances for a particular value of L and for a particular heuristic.

Comparing the obtained results for HS1 with the results in Hasani *et al.* [8] for the Min-idle algorithm which we denote as (Mit) and the results in Abdekhodaee and Wirth [3] for forward heuristic which we denote as (FH) and also comparing the obtained results for HS2 with the results in Hasani *et al.* [8] for the Min-loadgap algorithm which we denote as (Mlg) and the results in Abdekhodaee and Wirth [3] for backward heuristic which we denote as (BH) when using the same instances, the following summary can be given:

In Table 3, HS1 is compared with FH and Mit for $L \in \{0.1, 0.5, 0.8\}$ and for $m = 2$ since FH and Mit can be applied to only the case of two machines. It must be noted that FH and Mit provide the same result for all proposed instances. With R_2 (LPT) rule and for the case of $L = 0.1$ HS1 outperform by far the two heuristics FH and Mit for the majority of cases. For $L = 0.5$ and $L = 0.8$ FH and Mit are better than HS1 for the two proposed scheduling rules R_2 and R_4.

In Table 4, HS1 is compared with FH in the case of $L = 0.1$ for medium/large instances. It is observed that HS1 with the LPST rule is better than FH in term of deviation from the lower bound in the most of the data sets $n \in \{50, 100, 150, 200\}$. In some cases of $n \in \{250, 300, 350\}$, HS1 with the LST rule gives the same results in comparison with FH.

In Table 5, HS2 is compared with BH and Mlg for small size instances. The symbol $*$ is used to specify that no solution can be found since Mlg was proposed only for the case of two machines and for BH we adapted the algorithm for the case of arbitrary number of machines. With R_2 (LPT) rule and for the case of $m \in \{2, 3, 4, 5\}$ HS2 is better than BH in term of deviation from the lower bound for some cases for $m \geq 2$ and also it gives less precise results in comparison with Mlg.

In Table 6, HS2 is compared with BH for medium/large instances. With LPT rule and for $m \in \{3, 4, 5\}$, HS2 outperform BH in term of deviation from the lower bound for the most cases and for all values of the server load $L \in \{1.5, 1.8, 2\}$. Therefore, with LST rule the heuristic HS2 gave less precise results in comparison with BH and with the R_2 rule also. It must be noted that the solution time with all proposed heuristics (HS1, HS2, BH, FH, Mit, Mlg) and for all proposed instances don't exceed 0,006 seconds.

n	m		$L = 0.1$			$L = 0.5$			$L = 0.8$		
			HS1-LPT	HS1-LST	FH / (Mit)	HS1-LPT	HS1-LST	FH / (Mit)	HS1-LPT	HS1-LST	FH / (Mit)
8	2	R_{avg}	1.01413	1.03591	1.03591	1.11347	1.17261	1.08607	1.17999	1.20681	1.13497
		R_{max}	1.03659	1.11385	1.11385	1.19561	1.30882	1.13388	1.34456	1.35988	1.17917
20	2	R_{avg}	1.0132	1.02269	1.02464	1.08481	1.06107	1.04708	1.19439	1.08417	1.1065
		R_{max}	1.02339	1.03768	1.04288	1.12479	1.13828	1.09338	1.26943	1.17402	1.27185
50	2	R_{avg}	1.00423	1.01033	1.01119	1.05972	1.0205	1.01431	1.14087	1.06299	1.03064
		R_{max}	1.00779	1.03451	1.0308	1.077	1.03686	1.02459	1.17022	1.12111	1.0761
100	2	R_{avg}	1.00341	1.00611	1.00611	1.06456	1.01405	1.0091	1.15299	1.04793	1.02527
		R_{max}	1.00821	1.01296	1.01296	1.08852	1.04478	1.03046	1.18127	1.08919	1.04048
150	2	R_{avg}	1.00539	1.00459	1.00459	1.06214	1.00572	1.00429	1.1301	1.01687	1.00722
		R_{max}	1.00953	1.00745	1.00745	1.08458	1.01021	1.00822	1.1466	1.03687	1.01504
200	2	R_{avg}	1.004	1.00484	1.00484	1.06572	1.00443	1.00259	1.13859	1.01611	1.00844
		R_{max}	1.00564	1.00825	1.00825	1.07941	1.00985	1.00518	1.15395	1.02805	1.01666
250	2	R_{avg}	1.00425	1.00251	1.00251	1.06451	1.0037	1.0023	1.13322	1.01129	1.00498
		R_{max}	1.00534	1.00614	1.00614	1.08621	1.00883	1.0061	1.15714	1.0228	1.01124
300	2	R_{avg}	1.00478	1.00182	1.00182	1.06345	1.00144	1.00098	1.14009	1.02038	1.00678
		R_{max}	1.00699	1.0047	1.0047	1.07094	1.00509	1.00262	1.14519	1.03069	1.01391
350	2	R_{avg}	1.00404	1.0009	1.0009	1.06402	1.00122	1.00098	1.13749	1.01518	1.00646
		R_{max}	1.00532	1.00248	1.00248	1.07527	1.00395	1.00167	1.17283	1.03719	1.02185

Table 3: Computational results with HS1 heuristic for small/medium and large-sizes instances

n	m		$L = 0.1$						
			HS1-SST	HS1-LST	HS1-SPT	HS1-LPT	HS1-SPST	HS1-LPST	FH
50	2	R_{avg}	1.01833	1.01033	1.02	1.00423	1.02074	1.00391	1.01119
		R_{max}	1.03776	1.03451	1.03127	1.00779	1.03253	1.00928	1.0308
100	2	R_{avg}	1.01191	1.00611	1.00931	1.00341	1.01007	1.00235	1.00611
		R_{max}	1.02391	1.01296	1.01621	1.00821	1.0139	1.00602	1.01296
150	2	R_{avg}	1.01008	1.00459	1.00787	1.00539	1.0065	1.0045	1.00459
		R_{max}	1.0144	1.00745	1.01156	1.00953	1.00798	1.008	1.00745
200	2	R_{avg}	1.0109	1.00484	1.0057	1.004	1.00459	1.00337	1.00484
		R_{max}	1.01831	1.00825	1.00857	1.00564	1.00582	1.0044	1.00825
250	2	R_{avg}	1.00831	1.00251	1.00391	1.00425	1.00377	1.00341	1.00251
		R_{max}	1.01483	1.00614	1.00686	1.00534	1.00494	1.00463	1.00614
300	2	R_{avg}	1.00829	1.00182	1.00303	1.00478	1.00318	1.00404	1.00182
		R_{max}	1.01161	1.0047	1.00527	1.00699	1.00431	1.00624	1.0047
350	2	R_{avg}	1.00723	1.0009	1.00339	1.00404	1.0027	1.00327	1.0009
		R_{max}	1.00994	1.00248	1.00546	1.00532	1.00374	1.00433	1.00248

Table 4: Computational results with HS1 heuristic for medium and large-sizes instances for $L = 0.1$

n	m		$L = 1.5$			$L = 1.8$			$L = 2$		
			HS2-LPT	BH	Mlg	HS2-LPT	BH	Mlg	HS2-LPT	BH	Mlg
8	2	R_{avg}	1.04044	1.02051	1.03388	1.03848	1.0274	1.03959	1.03309	1.02827	1.0348
		R_{max}	1.16529	1.08058	1.16942	1.11372	1.11372	1.11045	1.09957	1.09668	1.13131
	3	R_{avg}	1	1	*	1.00103	1.01303	*	1.00101	1.0206	*
		R_{max}	1	1	*	1.00266	1.0625	*	1.00504	1.05804	*
20	2	R_{avg}	1.02533	1.02085	1.01949	1.044	1.03147	1.05137	1.01054	1.00295	1.00536
		R_{max}	1.039	1.03874	1.04212	1.09224	1.07453	1.09835	1.03893	1.01416	1.02679
	3	R_{avg}	1.00045	1.00076	*	1.00011	1.00011	*	1.0001	1.0001	*
		R_{max}	1.00226	1.00226	*	1.00057	1.00057	*	1.0005	1.0005	*
	4	R_{avg}	1.00083	1.00547	*	1	1	*	1	1	*
		R_{max}	1.00413	1.02324	*	1	1	*	1	1	*
	5	R_{avg}	1	1	*	1	1	*	1.00011	1.00011	*
		R_{max}	1	1	*	1	1	*	1.00054	1.00054	*

Table 5: Computational results with HS2 heuristic for small-sizes instances

5 Conclusion

In this contribution, the problem $Pm, S1|s_j, p_j|C_{max}$ was considered. Two greedy heuristics were proposed to solve this problem for small, medium and large instances. The first one was developed for instances with $LB = LB1$, and the second one for instances with $LB = LB2$. The heuristics performed extremely well in terms of the deviation from the proposed lower bounds for small/medium and large instances and also in terms of the low computational times. A performance comparison was performed between these two heuristics and other heuristics proposed in the literature for the case of two machines and also an arbitrary number of machines. The comparison shows that our heuristic HS1 outperform the literature in term of deviation from the lower bound in some cases. And the heuristic HS2 outperforms by far the literature for the case of arbitrary number of machines.

In future work, it is intended to investigate several types of metaheuristics for the parallel machine scheduling problem with single server, and also take into consideration other types of objective functions such as the total completion time or the total tardiness.

References

[1] S.A. Kravchenko and F. Werner, Parallel machine scheduling problems with a single server, Mathemati-

n	m		$L = 1.5$			$L = 1.8$			$L = 2$		
			HS2-LST	HS2-LPT	BH	HS2-LST	HS2-LPT	BH	HS2-LST	HS2-LPT	BH
30	3	R_{avg}	1.03675	1	1.00264	1.01881	1	1.00026	1.01463	1	1
		R_{max}	1.0521	1	1.01634	1.03632	1	1.0026	1.03206	1	1
	4	R_{avg}	1.01587	1.00014	1.00014	1.01111	1.00018	1.00018	1.00667	1	1
		R_{max}	1.0427	1.00141	1.00141	1.03298	1.00181	1.00181	1.01518	1	1
40	3	R_{avg}	1.03217	1	1.0025	1.01793	1	1	1.02601	1.00034	1.0007
		R_{max}	1.06977	1	1.0184	1.04368	1	1	1.04756	1.00336	1.00336
	4	R_{avg}	1.01507	1	1	1.01129	1	1	1.00706	1	1
		R_{max}	1.02889	1	1	1.03012	1	1	1.02292	1	1
	5	R_{avg}	1.0104	1	1	1.00475	1	1	1.00434	1	1
		R_{max}	1.04315	1	1	1.01673	1	1	1.01656	1	1
50	3	R_{avg}	1.03099	1	1.00145	1.01656	1	1.00007	1.01139	1	1
		R_{max}	1.06379	1	1.00848	1.03549	1	1.00065	1.02464	1	1
	4	R_{avg}	1.01589	1	1	1.00885	1	1	1.01021	1	1
		R_{max}	1.03005	1	1	1.01821	1	1	1.02041	1	1
60	3	R_{avg}	1.03488	1.00002	1.00027	1.02079	1	1.00029	1.01353	1	1
		R_{max}	1.05075	1.00021	1.00142	1.03311	1	1.00235	1.0275	1	1
	5	R_{avg}	1.00866	1.00002	1.00002	1.00591	1	1	1.00388	1	1
		R_{max}	1.01784	1.00021	1.00021	1.01442	1	1	1.01184	1	1
100	3	R_{avg}	1.02453	1	1	1.0201	1	1.0001	1.01582	1	1
		R_{max}	1.03763	1	1	1.03738	1	1.00102	1.02496	1	1
	5	R_{avg}	1.00743	1	1	1.00481	1	1	1.00416	1	1
		R_{max}	1.01704	1	1	1.00885	1	1	1.01072	1	1
200	3	R_{avg}	1.02827	1	1.00002	1.01729	1	1	1.01626	1	1
		R_{max}	1.0397	1	1.00014	1.02506	1	1	1.02066	1	1.00005
	4	R_{avg}	1.01364	1	1	1.00945	1	1	1.00784	1	1
		R_{max}	1.02029	1	1	1.01339	1	1	1.01123	1	1
250	3	R_{avg}	1.02587	1	1.00006	1.01829	1	1.00001	1.01581	1	1
		R_{max}	1.04181	1	1.00051	1.03103	1	1.00009	1.0214	1	1
	5	R_{avg}	1.0069	1	1	1.00549	1	1	1.00513	1	1
		R_{max}	1.01086	1	1	1.00847	1	1	1.00819	1	1

Table 6: Computational results with HS2 heuristic for medium and large-sizes instances

cal and Computer Modeling, 26, 1-11, 1997.

[2] P. Brucker, C. Dhaenens-Flipo, S. Knust, S. Kravchenko, A. Svetlana and F. Werner, Complexity results for parallel machine problems with a single server, Journal of Scheduling, 5, 429-457, 2002.

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

[4] A.H. Abdekhodaee, A. Wirth and H.S. Gan, Equal processing and equal setup time cases of scheduling parallel machines with a single server, Computers & Operations Research, 31, 1867-1889, 2004.

[5] A.H. Abdekhodaee, A. Wirth and H.S. Gan, Scheduling two parallel machines with a single server: the general case, Computers & Operations Research, 33, 994-1009, 2006.

[6] H. S. Gan, A. Wirth and A.H. Abdekhodaee, A branch-and-price algorithm for the general case of scheduling parallel machines with a single server, Computers & Operations Research, 39 (9), 2242-2247, 2012.

[7] K. Hasani, S. A. Kravchenko and F. Werner, Simulated annealing and genetic algorithms for the two-machine scheduling problem with a single server, International Journal of Production Research, 52 (13), 3778-3792, 2014.

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

[9] J. P. Arnaout, Heuristics for the two machine scheduling problem with a single server, International Transactions in Operational Research, 24 (6), 1347-1355, 2017.

[10] M.Y. Kim and Y.H. Lee, MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server, Computers & Operations Research, 39, 2457-2468, 2012.

[11] A. El idrissi, R. Benmansour, M. Benbrahim, D. Duviver, MIP formulations for identical parallel machine scheduling problem with single server, Proceedings of 4th IEEE International Conference on Optimization and Applications (ICOA), 2018.

[12] C.P. Koulamas, M.L. Smith, Look-ahead scheduling for minimizing machine interference, International Journal of Production Research, 26 (9), 1523-1533, 1988.

[13] R. Benmansour, O. Braun, H. Allaoui, Modeling the single processor scheduling problem with time restrictions as a parallel machine scheduling problem, Proceeding of 7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), 2015.