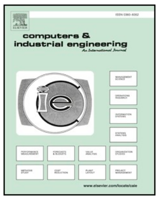
Computers & Industrial Engineering 140 (2020) 106223

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie



# New efficient constructive heuristics for the two-stage multi-machine assembly scheduling problem

Carla Talens[*], Victor Fernandez-Viagas, Paz Perez-Gonzalez, Jose M. Framinan

*Industrial Management, School of Engineering, University of Seville, Spain*

ARTICLE INFO

*Keywords:*
Scheduling
Assembly
Total completion time
Constructive heuristics

ABSTRACT

In this paper, we address the two-stage multi-machine assembly scheduling problem, a layout consisting of several dedicated parallel machines in the first stage and identical parallel machines in the second stage. The objective considered is the minimisation of the total completion time. Despite its relevance in practice and its NP-hard nature, this problem has not received much attention so far in the literature. In our paper, we propose two new efficient heuristics. The first heuristic constructs a solution taking into account some specific knowledge of the problem domain. This algorithm is embedded into a beam search-based constructive heuristic and its behaviour when the beam width takes different values is analysed. The computational experiments carried out show that the proposals are more efficient than the existing heuristics for the problem and also the adaptations of heuristics from related problems.

## 1. Introduction

Assembly scheduling problems have many applications in industry since many products are made up of different components that need to be manufactured in the earlier stages and then assembled into final products in a later stage. These decision problems are receiving an increasing attention of researchers due to its applications in industry (Sheikh, Komaki, & Kayvanfar, 2018), such as personal computer manufacturing (Potts, Sevast'janov, Strusevich, Van Wassenhove, & Zwaneveld, 1995), fire engine assembly plants (Lee, Cheng, & Lin, 1993), circuit board production (Cheng & Wang, 1999), food and fertilizer production (Hwang & Lin, 2012), car assembly industry (Fattahi, Hosseini, & Jolai, 2013), motor assembly industry (Liao, Lee, & Lee, 2015), or plastic industry (Allahverdi & Aydilek, 2015). Other examples can also be found in services/IT, including distributed database systems (Allahverdi & Al-Anzi, 2006; Al-Anzi & Allahverdi, 2006; Al-Anzi & Allahverdi, 2007), or multi-page invoice printing systems (Zhang, Zhou, & Liu, 2010).

In this paper we focus on a two-stage multi-machine assembly scheduling problem where there are several dedicated machines in the first stage (e.g. to manufacture the different components of the final product) and more than one identical (parallel) machines in the second stage (e.g. to assembly the components into the final product). Clearly, processing a job in the second stage –which can be done by any of the parallel machines– can start only after all operations for this job in the first stage have been completed. The objective is to minimize the total completion time so, according to the classification and notation in Framinan, Perez-Gonzalez, and Fernandez-Viagas (2019), it can be denoted as the $DP_m \rightarrow P_m || \sum_j C_j$ problem. This problem can be seen as a generalisation of the two-machine flowshop problem since, for the case with only one machine at the first stage and one assembly machine at the second stage (Jung, Woo, & Kim, 2017), it is equivalent to scheduling in a flowshop with two machines (Sung & Kim, 2008; Framinan & Perez-Gonzalez, 2017). Then, since Gonzalez and Sahni (1978) proved that the $Fm || \sum_j C_j$ is NP-hard in the strong sense, it is clear that the problem under study is strongly NP-hard.

While the two-stage assembly problem with only one machine in the second stage has been widely discussed in the literature (see the recent review on the topic by Framinan et al. (2019)), to the best of our knowledge, the problem considering several (identical) assembly machines has received much less attention. Furthermore, most approximate algorithms for this problem consider at maximum two assembly machines in the second stage in their computational experience, and do not compare their performance with existing algorithms for related problems, more notably with algorithms for the assembly scheduling problem with one machine in the assembly stage, or the customer order scheduling problem (which can be seen as an assembly scheduling problem with zero processing times in the assembly stage, see Framinan et al. (2019)). Therefore, the main contribution of this paper is to improve the performance of the approximate solution procedures for the

[*] Corresponding author at: Ave. Descubrimientos s/n. 41092 Seville, Spain.
*E-mail address:* cartafa@us.es (C. Talens).

https://doi.org/10.1016/j.cie.2019.106223
Received 17 April 2019; Received in revised form 28 October 2019; Accepted 10 December 2019
Available online 16 December 2019

problem under consideration by proposing two new constructive heuristics that explicitly use knowledge of the problem domain to construct the solutions. More specifically, we propose a fast constructive heuristic which applies to our problem some dominance properties by Framinan and Perez-Gonzalez (2017) derived for the case where there is only one assembly machine. In light of the excellent results of this heuristic in negligible computation times, it is embedded into a beam search-based constructive heuristic. This type of heuristic has been recently applied by Fernandez-Viagas and Framinan (2017, 2016) and Fernandez-Viagas, Valente, and Framinan (2018) for related flowshop scheduling problems. In addition, we introduce the idea of keeping only the most promising nodes in each iteration to boost its performance, an idea successfully employed for different scheduling problems (see e.g. Della Croce & T'kindt (2003), Valente & Alves (2008), and Valente (2010)). An extensive computational evaluation is performed to show that the proposals are found to be more efficient than existing solution procedures for the problem.

The remainder of the paper is organised as follows: the problem is formally described and the state of the art is presented in Section 2. In Section 3, we explain in detail the proposed heuristics: The fast constructive heuristic is presented in Section 3.1, while in Section 3.2 the beam search heuristic is explained. The computational experiments are conducted in Section 4 and, finally, conclusions are discussed in Section 5.

## 2. Problem statement and background

The problem studied in this paper can be stated as follows: There are $n$ jobs to be scheduled in a layout composed of two stages. Each job has $m_1 + 1$ operations. In the first stage, there are dedicated parallel machines, in which the first $m_1$ operations are conducted, while in the assembly stage there are $m_2$ identical parallel machines. Only after all $m_1$ operations are completed, the assembly operation may start in the first available machine of the second stage. A job $j$ has a processing time $p_{ij}$ on machine $i$ in the first stage and an assembly processing time $at_j$ in the second stage. The decision problem consists on scheduling the jobs in the two machines so the sum of the completion times of the jobs is minimised.

A solution for this problem can be given by giving a sequence of jobs indicating the order in which the jobs are processed (see e..g. Al-Anzi & Allahverdi (2006) and Al-Anzi & Allahverdi (2012)). Therefore, given a sequence, let $[j]$ denote the job processed in position $j$ in the sequence. $C1_j$ the maximum completion time of job in position $j$ in the first stage can be computed as follows:

$$C1_j = \max_{i=1,\ldots,m_1} \left\{ \sum_{k=1}^{j} p_{i[k]} \right\} \tag{1}$$

The completion time of each job in position $j$ in the sequence is computed by Eq. (2) recursively for each assembly machine $i = 1\ldots,m_2$ using the variable $C_{ij}$ to denote the completion time in machine $i$ of the jobs in the sequence until position $j$:

$$C_{ij} = \begin{cases} \max\{C_{i^*j-1}, C1_j + at_{[j]}\} & \text{if } i = i^* \text{with } i^* = \text{argmin}_{i=1,\cdots,m_2}\{C_{i,j-1}\} \\ C_{i,j-1} & \text{otherwise} \end{cases} \tag{2}$$

where $C_{i0} = 0 \quad \forall\ i = 1\ldots m_2$. Then, $C_j$ the completion time of the job processed in position $j$ of the sequence can be computed by Eq. (3).

$$C_j = C_{i^*j} \quad \forall\ i = 1\ldots m_2, \quad \forall\ j \tag{3}$$

As mentioned in Section 1, there are few references addressing the assembly scheduling problem with several machines in the second stage, namely the works by Sung and Kim (2008) and Al-Anzi and Allahverdi (2012). Sung and Kim (2008) develop an heuristic, denoted *SAK* from now on, applying a processing-time-based pairwise exchange mechanism, while in Al-Anzi and Allahverdi (2012), a mathematical model of the problem with two assembly machines is proposed, together with three new metaheuristics.

In view of the relationship of the problem under consideration with other scheduling problems, it is worth also to analyse the existing solution procedures for related problems, namely for the assembly scheduling problem with one machine in the last stage ($DP_m \rightarrow 1||\sum C_j$ problem) and the customer order scheduling problem ($DP_m \rightarrow 0||\sum C_j$ problem).

Regarding the $DP_m \rightarrow 1||\sum C_j$ problem, the first reference addressing it is Tozkapan, Kirca, and Chung (2003), where the authors prove that permutation schedules are optimal for the $DP_m \rightarrow 1$ problem and propose two heuristics, labelled $TCK1$ and $TCK2$ in the following, to find an upper bound for their branch and bound algorithm. Al-Anzi and Allahverdi (2006) also address this problem and derive a number of theoretical properties. They propose three simple constructive heuristics ($S1$, $S2$ and $S3$) based on the idea of ordering the jobs according to the Shortest Processing Time (SPT) rule, and two additional constructive heuristics, labelled $A1$ and $A2$ in the following. Recently, Framinan and Perez-Gonzalez (2017) develop a constructive heuristic, denoted FAP in the following, which outperforms the existing constructive heuristics and is based on the problem properties studied by Al-Anzi and Allahverdi (2006). The last reference where this problem is considered is Lee (2018). Six lower bounds are proposed and tested in a branch and bound algorithm. The author also proposes four greedy-type constructive heuristics, labelled $G1$, $G2$, $G3$ and $G4$.

The assembly scheduling problem with a single assembly machine has been tackled with respect to different objectives. Lee et al. (1993) and Potts et al. (1995) addressed this problem regarding the minimisation of the makespan. Other papers address different objectives such as the minimisation of the maximum lateness (Al-Anzi & Allahverdi (2006) and Allahverdi & Al-Anzi (2006)), additional constraints such as setup times (Al-Anzi & Allahverdi (2007)), or additional stages for the transportation of components (Koulamas & Kyparisis (2001) and Shoaardebili & Fattahi (2015)).

Regarding the $DP_m \rightarrow 0||\sum C_j$ problem, note that this problem is tantamount to the one under consideration if the processing times of the jobs in the assembly stage are zero. Even if this is not the usual case, both problems could be very similar also if the processing times in the first stage are much higher than in the assembly stage. Therefore, solution methods for the customer order scheduling problem could potentially be applied to our problem. For the $DP_m \rightarrow 0||\sum C_j$ problem, Sung and Yoon (1998) propose two constructive heuristics based on the SPT rule. The first one schedules the order with the smallest total processing time across all $m$ machines, labelled *STPT* in the following, and the second one selects the order with the smallest maximum amount of processing time on any of the $m$ machines, denoted as *SMPT*. Leung, Li, and Pinedo (2005) propose a constructive heuristic that selects as the next order to be sequenced the one that would be completed the earliest, that is, the order with the Earliest Completion Time (ECT). Based on this idea and including some look-ahead concepts, Framinan and Perez-Gonzalez (2017) propose a constructive heuristic and two specific local search mechanisms for the problem, labelled $SHIFT_k$ and $SHIFT_{kOPT}$.

After this review, it can be seen that, despite some solution procedures exist for the problem, the performance of the adaptation of procedures from related problems has not been tested so far. Furthermore, there is some opportunity to improve existing methods for the problem by incorporating some knowledge of the problem domain. These methods are presented in the next section.

## 3. Proposed constructive heuristics

In this section, we propose two heuristics for this problem. First, a constructive heuristic is detailed (Section 3.1) and then we propose a beam search based constructive heuristic (Section 3.2).

```
Procedure Proposed Constructive Heuristic CH
    // All jobs are initially unscheduled
    $\Pi := \varnothing$;
    // Completion times on stages 1 and 2:
    $C1_i := 0 \quad i = 1, \ldots, m_1$
    $C2_i := 0 \quad i = 1, \ldots, m_2$
    $i^* := \arg\min_{1 \le i \le m_2} C2_i$;
    Obtain a sequence $\mathcal{U} := \{\omega_1, \ldots, \omega_n\}$ by applying algorithm S2;
    for $j = 1$ to $n$ do
        for each $\omega_j \in \mathcal{U}$ do
            // Compute the completion times in the first stage after selecting $\omega_j$ as candidate:
            $C_1(\omega_j) := \max_{1 \le i \le m_1}\{C1_i + p_{i\omega_j}\}$
            // Compute the idle time induced if job $\omega_j$ is inserted at the end of the partial sequence:
            $IT_j = \max\{C_1(\omega_j) - C2_i^*, 0\}$
            // Compute the expected contribution to the total completion time induced when job $\omega_j$ is inserted at the end of the partial sequence:
            $CT_j = \frac{at_{w_j}}{m_1 \cdot m_2}$
            // Compute $psi_j$:
            $\psi_j := a \cdot IT_j + CT_j$
        end
        $r := \arg\min_{1 \le k \le n-j+1} \psi_k$;
        Append $\omega_r$ at the end of $\Pi$, i.e. $\Pi := (\pi_1, \ldots, \pi_{j-1}, \omega_r)$;
        Extract $\omega_r$ from $\mathcal{U}$, i.e. $\mathcal{U} := \{\omega_1, \ldots, \omega_{r-1}, \omega_{r+1}, \ldots, \omega_{n-j+1}\}$;
        // Update values of the constructive sequence:
        $C1_i := C1_i + p_{i\omega_r}$
        $C2_i := \max\{C2_i^*, \max_{1 \le i \le m_1} C1_i\} + at_{w_r}$
    end
    return $C2_i$
end
```

**Fig. 1.** Pseudo-code of the proposed heuristic CH.

### 3.1. Fast constructive heuristic

The idea of the proposed heuristic – labelled $CH_{MMA}$ in the following– consists of iteratively constructing a sequence by selecting one job among the unscheduled jobs and adding it at the end of the partial sequence, an idea that has been addressed for different scheduling problems by Framinan and Perez-Gonzalez (2017, 2017) and Fernandez-Viagas and Framinan (2017) with excellent results. A key issue for the performance of this type of heuristics is the development of a problem-specific indicator $\psi$ which adequately represents the suitability of an unscheduled job to be appended since, once a job is added to the sequence, its position cannot be modified in the subsequent iterations.

Thence, the proposed algorithm starts with a set $\mathcal{U}$ containing all (unscheduled) jobs and an empty schedule $\mathcal{S}$. For each iteration $k \in \{1, \cdots, n\}$, each unscheduled job $\omega_j \in \mathcal{U}$ $(j = 1, \ldots, n - k + 1)$ is analysed as a candidate to be added to the last position in $S$, and its suitability is measured by computing the indicator $\psi_j$. Then, the job in $\mathcal{U}$ with the lowest value of $\psi_j$ is selected.

In our problem, two main aspects are considered to assess the suitability of appending a candidate job $\omega_j$ at the end of the partial sequence, i.e..

1. The case in which its inclusion implies that the first available machine in the second stage has to wait for processing the candidate job. Therefore, the idle time induced in the assembly stage by the insertion of job $\omega_j$ is computed. Let us $IT_j$ denote the idle time induced by scheduling job $\omega_j$ at the end of the sequence which is computed as follows:

$$IT_j = \max\{C1_j - (C_j - at_j), 0\} \tag{4}$$

Note that $C_j - at_j$ computes the workload of the first available machine in the second stage before scheduling job $j$ (i.e. the machine in the second stage where the candidate job will be processed). If the idle time induced by scheduling a job is greater than 0, then the completion times of the components in the first stage dominate the completion time of this job when it is evaluated as candidate to be scheduled. Otherwise, the completion time of the candidate job is largely influenced by the second stage. As we are minimising the total completion time, it is clear that the lower the idle time caused

```
Procedure $BSCH\_MMA(x)$
    Obtain a sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by applying algorithm S2;
    Update $\mathcal{U}_1^l(u_{1,1}^l = \omega_l)$ and $\mathcal{S}_1^l(s_{1,1}^l = \varnothing)$.
    for $l = 1$ to $x$ do
        $F_{1,l} = \psi_{\omega[l],0,l}$
    end
    for $k = 2$ to $n$ do
        // Candidates Nodes Creation
        Determination of $IT_{jkl}$, $CT_{jkl}$;
        // Candidates Nodes Evaluation
        $B_{jkl} := F_{kl} + a' \cdot IT_{jkl} + CT_{jkl}\ \forall\ l = 1, \cdots, x$ and $\forall\ j = 1, \cdots, n-k+1$;
        // Candidates Nodes Selection
        for $l' = 1$ to $x$ do
            Determination of the $l'$-th best candidate node according to non-decreasing $B_{jkl}$
            in iteration $k$. Denote by $branch[l']$ and $job[l']$ the value of $l$ and $j$ respectively
            of that candidate.
        end
        //Forecasting Phase.
        for $l' = 1$ to $x$ do
            Update $\mathcal{S}_{k+1}^{l'}$ and $\mathcal{U}_{k+1}^{l'}$ by removing job $u_{job[l'],k}^{branch[l']}$ from $\mathcal{U}_k^{branch[l']}$ and including
            in $\mathcal{S}_k^{branch[l']}$.
            $F_{k+1,l'} = F_{k,branch[l']} + b \cdot \psi_{job[l'],k,branch[l']}$;
        end
    end
    // Final evaluation
    Evaluate the total completion time of the scheduled jobs of each selected node and
    return the least one
end
```

**Fig. 2.** Pseudo-code of the proposed beam-search-based constructive heuristic.

by a job, the more suitable it is to be scheduled.

2. The contribution of the candidate job to the total completion time. As it can be seen, the idle time takes into account the suitability of the job until its processing in the second stage starts. In addition, its contribution to the total completion time would depend on the processing time in the second stage, i.e. $at_{w_j}$, which roughly measures the influence of the second stage. We weight this influence according to the number of assembly machines ($m_2$) since, with a higher number of machines in the second stage, the next jobs have a lower probability of not being affected by the second stage. Furthermore, to take into account the fact that, the higher the number of components in the first stage, the higher is its influence on the completion time, $CT_j$ the expected contribution of job $w_j$ to the total completion time is measured as follows:

$$CT_j = \frac{at_{w_j}}{m_1 \cdot m_2} \tag{5}$$

By taking into account these two aspects, we will ensure that the jobs to be first sequenced are those with lower values of idle time and assembly time. Therefore, the indicator $\psi_j$, which estimates the suitability of appending a candidate job $\omega_j$ at the end of $\mathcal{S}$, is computed as follows:

$$\psi_j = a \cdot IT_j + CT_j \tag{6}$$

where $a$ is a parameter to weight the influence of the two terms and that would be determined via calibration of the algorithm.

Note that the complexity of this heuristic is $O(n \cdot (n-k) \cdot m_1) \sim O(n^2 \cdot m_1)$, since the main loop in the algorithm performs $n$ iterations. In each iteration, $n-k$ jobs are evaluated, each evaluation consisting on obtaining the maximum processing time in the first stage, so each machine $m_1$ is evaluated. The pseudo-code of the proposed heuristic is shown in Fig. 1.

### 3.2. Beam search constructive heuristic

The heuristic proposed in Section 3.1 provides excellent results with negligible CPU times (see Section 4), proving that the indicator $\psi_j$ properly captures the suitability of a job to be appended. Therefore, we embed the components of this indicator into a new beam search-based constructive heuristic for the problem, labelled $BSCH_{MMA}$. This type of heuristic has been considered to solve different scheduling problems, such as in Sotskov, Tautenhahn, and Werner (1996) (where some constructive heuristics based on insertion techniques are combined with beam search for the permutation flowshop scheduling problem), Erenay, Sabuncuoglu, Toptal, and Tiwari (2010) (for the single machine bicriteria scheduling problem), and Fernandez-Viagas and Framinan (2017) (also for the permutation flowshop scheduling problem). In this type of heuristics, a number of candidate nodes, denoted by a parameter $x$, are maintained in each iteration. In iteration $k$, each node $l$ ($l \in \{1, \cdots, x\}$) is formed by a set of $k$ scheduled jobs, denoted as partial sequence $\mathcal{S}_k^l$, and a set of unscheduled jobs, $\mathcal{U}_k^l$ with $n-k+1$ jobs. Then, all unscheduled jobs in $\mathcal{U}_k^l$ are inserted in position $k+1$ of $\mathcal{S}_k^l$, thus obtaining $x \cdot (n-k+1)$ candidate nodes. Out of these nodes, the $x$ most suitable ones are selected as candidates for the next iteration. To deal with that, the ideas behind the indicator $\psi_j$ could be used in this heuristic. However, an additional complication arises because candidates from different nodes may have to be compared. More specifically, the heuristic may have to deal with one of the following situations:

- If the candidate jobs have been obtained by appending different jobs in $\mathcal{U}_k^l$ to the same node $l$, their partial sequences $\mathcal{S}_k^l$ will be exactly alike with the exception of the last job appended. So, the comparison can be done in reference to the completion time or the idle time caused by the added job.
- If the candidate jobs have been obtained from different nodes, the unscheduled jobs and the scheduled jobs are different for each candidate node. In this case, the comparison should take into account that the previous scheduled jobs at each candidate node are different, so this fact has to be considered when developing the indicator for suitability.

To explain the design of the heuristic in detail, we first denote by $\mathfrak{s}_{jk}^l$ the $j$th scheduled job of node $l$ in iteration $k$ and by $\mathfrak{u}_{jk}^l$ the $j$th unscheduled job of selected node $l$ in iteration $k$. As shown in Fig. 2, the heuristic consists of the following steps:

Step 1: Generate the initial $x$ nodes: All jobs are initially sorted according to Algorithm S2 by Al-Anzi and Allahverdi (2006), as it is done in $CH_{MMA}$. The first $x$ nodes are obtained by assigning the job in position $l$ of the sorted list to the first position of the partial sequence $\mathfrak{s}_{11}^l$ of the selected node $l$. The list of unscheduled jobs of this selected node $l$ is formed by the rest of the jobs.

Step 2: Generate candidate nodes: At iteration $k$, $n-k+1$ candidate jobs are obtained by appending each job in $\mathcal{U}_k^l$ at the end of the partial sequence of each selected node $l \in \{1, \cdots, x\}$.

Step 3: Evaluate candidate nodes: In this step, two aspects are considered: first, the influence from the selected node and, second, the influence from the inserted job. The former is computed as the forecast index, $F_{kl}$, which is explained in Step 5, and the latter is due to the insertion of the new job, $\mathfrak{u}_{jk}^l$, at the end of the partial sequence, which is measured by $CT_{jkl}$, analogously as Eq. (5), and by $IT_{jkl}$, which denotes the idle time incurred when inserting job $\mathfrak{u}_{jk}^l$ in the selected node and is computed analogously as Eq. (4). Note that these two last components are taken from the heuristic in Section 3.1, while $F_{kl}$ is a component specifically designed to allow the comparison of candidates from different nodes.
Therefore, at each iteration $k$, the following indicator is used to compute the suitability of appending an unscheduled job $\mathfrak{u}_{jk}^l$ in a selected node $l$:

$$B_{jkl} := F_{kl} + a' \cdot IT_{jkl} + CT_{jkl} \tag{7}$$

In Eq. (7) the parameter $a'$ has been considered in order to balance the idle time and the completion time of the new inserted job and its calibration is addressed in Section 4.3.

Step 4: Select the best $x$ candidate nodes: The $x$ candidate nodes with the lowest values of $B$ are selected and these nodes will form the nodes of the next iteration, i.e. in iteration $k$ all the combinations of $j$ and $l$ are tested and those achieving the lowest values of $B_{jkl}$, as defined in Eq. (7), are selected. The rest of candidate nodes are discarded and the best candidate nodes are defined as the selected nodes for the next iteration. At each iteration $k$, the combination of $l$ and $j$ of the $l'$th best $B_{jkl}$ ($l' \in \{1, \ldots, x\}$) are denoted by $branch[l']$ and $job[l']$, respectively.

Step 5: Update forecast index: The forecast index $F$ is defined in order to compare candidate nodes obtained from different nodes and, therefore, composed by different un- and scheduled jobs. $F$ represents the completion time of the last scheduled job at each candidate node and it is computed by Eq. (8):

$$F_{k,l'} = F_{k-1,branch[l']} + b \cdot \psi_{job[l'],k,branch[l']} \tag{8}$$

where parameter $b$ is designed to balance the influence of the last scheduled job in the completion time and $\psi_{job[l'],k,branch[l']}$ is defined by $\psi_{job[l'],k,branch[l']} = a' \cdot IT_j + CT_j$(similar as in Eq. (6)) and computed when job $\mathfrak{u}_{jk}^l$ is appended. The calibration of $b$ is discussed in Section 4.3. The pseudocode of the algorithm is shown in Fig. 2.
Apart from the weights $a'$ and $b$ (which are determined during the calibration of the algorithm), the $BSCH_{MMA}$ has only one parameter ($x$, the beam width). It can be seen that, for $x = 1$, $BSCH_{MMA}$ is tantamount to $CH_{MMA}$. Note that the complexity of this heuristic is $O(max\{n^2xm_1, n^2xm_2, n^2x^2\})$, since the main loop in the algorithm performs $n$ iterations. In each iteration, $n-k+1$ jobs are evaluated in each node $x$. This evaluation consists of obtaining the maximum processing time in the first stage, so each $m_1$ is evaluated (complexity $n^2xm_1$), and assigning the selected job to the first available assembly machine, so each $m_2$ is evaluated (complexity $n^2xm_2$). In addition, for each node $x$, the rest of nodes are evaluated in order to select the best $x$ combinations of nodes and jobs for the next iteration (complexity $n^2x^2$).

### 3.2.1. Variable beam width

To the best of our knowledge, beam search-based heuristics employed in the scheduling literature always use a constant beam width $x$. However, it is expected that the number of nodes analysed in each iteration has a large influence on the performance of this heuristic. To test this hypothesis, we carry out this study and analyse the behaviour of the $BSCH_{MMA}$ when $x$ may take different values. More specifically, we will test the following variants:

- Constant Beam Width: $BSCH_{MMA}$ is tested with different values of $x$. In this manner, the influence of the beam width on the performance of the heuristic can be analysed.
- Ascending Beam Width: In this version, the heuristic, denoted as $BSCH_{ASC}$, starts selecting $x$ nodes and the beam width increases in one unit as the beam search advances. The search is stronger on each iteration since the number of selected nodes is higher.
- Descending Beam Width: This version, labelled as $BSCH_{DESC}$, starts by selecting a number of nodes equal to $x+n-1$, and the beam width decreases one by one as the number of iterations increases. Therefore, in the last iteration, $x$ nodes are considered.
- V-shaped Beam Width: In this version, the beam width is modified taking a V-shape. Initially, $x$ nodes are considered and, for each iteration $k$, the number of nodes is decreased one by one while $k \leqslant \frac{n}{2}$ and then it increases until $k = n$. We denote this version as $BSCH_V$.
- Peak-shaped Beam Width: The pattern of this version, labelled as $BSCH_P$, is completely opposed to the previous one. The initial beam width is $x$, and then it increases one by one whereas $k \leqslant n \cdot \frac{2}{3}$ and then it decreases also one by one until the last iteration.

We design and implement different versions, which are evaluated considering the next values of $x$, $x \in \{2, 5, 10, 15, \frac{n}{10}, n, n + \frac{n}{2}, 2n\}$.

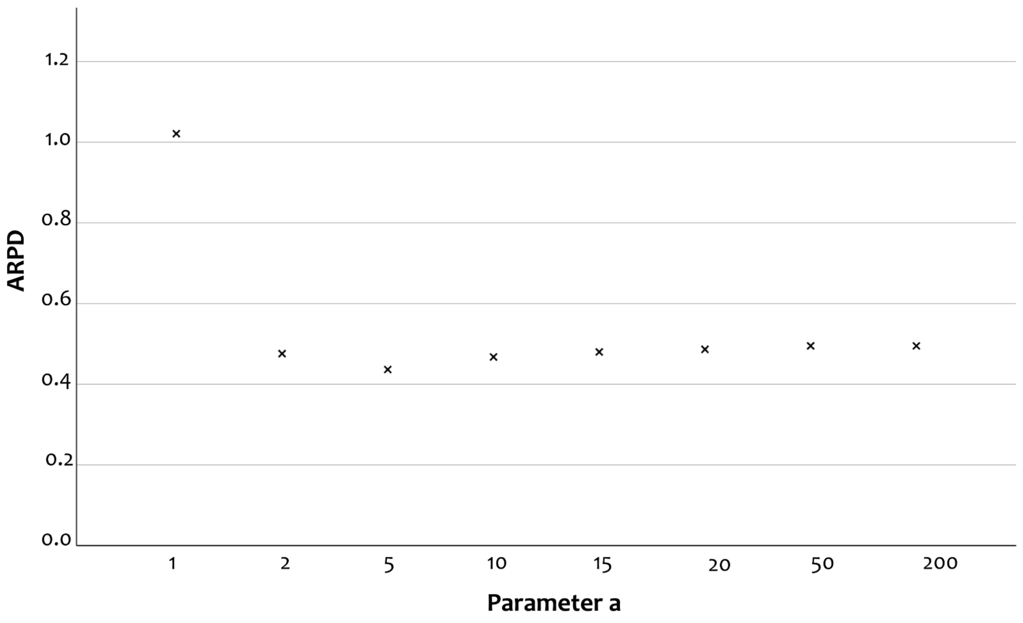


**Fig. 3.** Evolution of ARPD with the different values of parameter $a$.

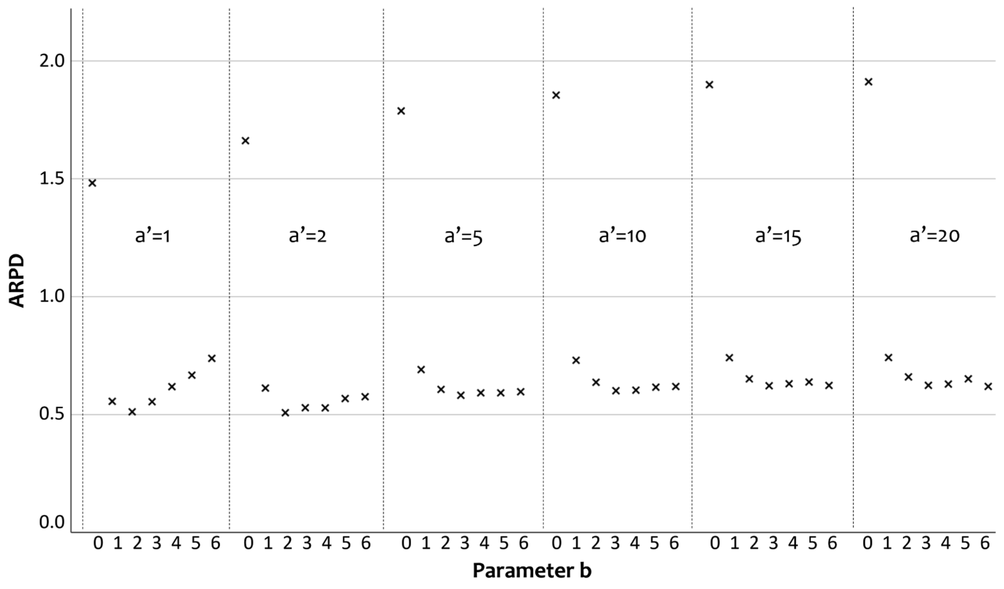


**Fig. 4.** Evolution of ARPD with the different values of parameters $a'$ and $b$.

**Table 1**
Holm's procedure for comparison of the different versions of *BSCH*.

| $i$ | $H_i$ | $p$-value | Wilcoxon | $\alpha/(k-i-1)$ | Holm's Procedure |
|---|---|---|---|---|---|
| 1 | $BSCH_{MMA}$ $(x = 10) = BSCH_{ASC}$ $(x = 2)$ | 0.015 | R | 0.0083 | – |
| 2 | $BSCH_{MMA}$ $(x = 5) = BSCH_V$ $(x = 2)$ | 0.001 | R | 0.0100 | R |
| 3 | $BSCH_{MMA}$ $(x = 5) = BSCH_P$ $(x = 2)$ | 0.000 | R | 0.0125 | R |
| 4 | $BSCH_{MMA}$ $(x = 10) = BSCH_P$ $(x = 5)$ | 0.000 | R | 0.0167 | R |
| 5 | $BSCH_{MMA}$ $(x = 10) = BSCH_V$ $(x = 2)$ | 0.025 | R | 0.0125 | R |
| 6 | $BSCH_{MMA}$ $(x = 15) = BSCH_{DESC}$ $(x = 2)$ | 0.000 | R | 0.0500 | R |

## 4. Computational evaluation

In this section, we analyse the efficiency of the heuristics proposed in Section 3. The indicators computed to compare the different algorithms are presented in Section 4.1. The testbed employed for the comparison is designed in Section 4.2, while in Section 4.3 we perform a design of experiments to set up proper values for parameters $a$ for $CH_{MMA}$, $a'$ and $b$ for $BSCH_{MMA}$. In Section 4.4, the different versions of the beam-search heuristics presented in Section 3.2.1 are compared to obtain the best variants. Finally, in Section 4.5 the proposed heuristics are compared with existing heuristics for the problem and for related problems. All methods have been coded in C# using Visual Studio and carried out in an Intel Core i7-3770 PC with 3.4 GHz and 16 GB RAM, using the same common functions and libraries. In order to obtain a

**Table 2**
Summary of results of the different versions of *BSCH*.

| $x$ | 2 | 5 | $n/10$ | 10 | 15 | $n$ | $n + n/2$ | $n$ |
|---|---|---|---|---|---|---|---|---|
| | | | | *ARPD* | | | | |
| $BSCH_{MMA}$ | 1.3196 | 0.7708 | 0.8064 | 0.6437 | 0.5874 | 0.5382 | 0.5227 | 0.5408 |
| $BSCH_{ASC}$ | 0.5499 | 0.5467 | 0.5360 | 0.5423 | 0.5524 | 0.5282 | 0.5452 | 0.5568 |
| $BSCH_{DESC}$ | 0.5499 | 0.5467 | 0.5360 | 0.5423 | 0.5436 | 0.5543 | 0.5736 | 0.5790 |
| $BSCH_V$ | 0.5458 | 0.5324 | 0.5295 | 0.5349 | 0.5359 | 0.5425 | 0.5576 | 0.5646 |
| $BSCH_P$ | 0.6174 | 0.5777 | 0.5825 | 0.5626 | 0.5540 | 0.5348 | 0.5498 | 0.5617 |
| | | | | *ACPU* | | | | |
| $BSCH_{MMA}$ | 0.0040 | 0.0104 | 0.0173 | 0.0279 | 0.0430 | 0.1220 | 0.2685 | 0.5026 |
| $BSCH_{ASC}$ | 0.0268 | 0.0568 | 0.0874 | 0.1229 | 0.1648 | 0.2803 | 0.4638 | 0.7294 |
| $BSCH_{DESC}$ | 0.0439 | 0.0916 | 0.1406 | 0.1962 | 0.2602 | 0.4189 | 0.6593 | 0.9969 |
| $BSCH_V$ | 0.0251 | 0.0522 | 0.0796 | 0.1100 | 0.1442 | 0.2199 | 0.3305 | 0.4821 |
| $BSCH_P$ | 0.6482 | 0.6084 | 0.6133 | 0.5934 | 0.5848 | 0.5656 | 0.5806 | 0.5924 |

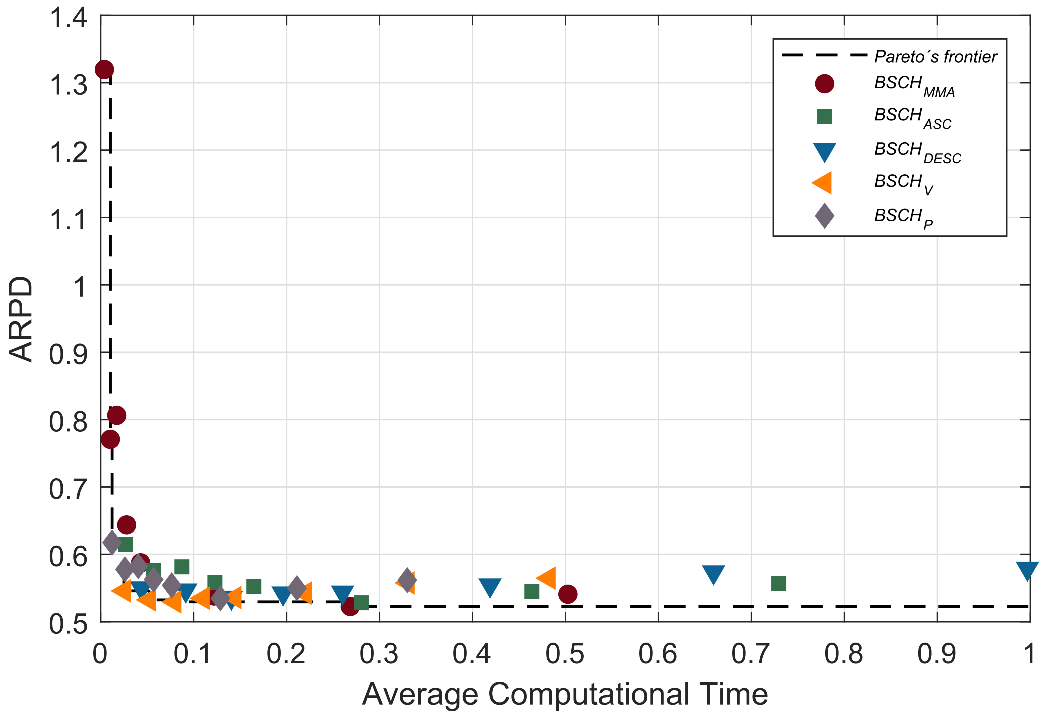


**Fig. 5.** ARPD versus average CPU times of the different versions of *BSCH* with the Pareto frontier.

**Table 3**
Summary of results of the different heuristics.

| Heuristic | *ARPD* | *ACPU* | *ARPT* | Heuristic | *ARPD* | *ACPU* | *ARPT* |
|---|---|---|---|---|---|---|---|
| $CH_{MMA}$ | 1.8742 | 0.001901 | 625.07 | *SAK* | 12.6792 | 0.359579 | 102534.03 |
| *FAP* | 1.6644 | 0.011425 | 3358.46 | *STPT* | 15.8731 | 0.000005 | 1.64 |
| *G*1 | 17.4028 | 0.001870 | 615.74 | *SMPT* | 21.6706 | 0.000007 | 2.36 |
| *G*2 | 6.1421 | 0.001845 | 605.78 | *ECT* | 17.4028 | 0.036756 | 10559.23 |
| *G*3 | 6.8172 | 0.001799 | 590.66 | $SHIFT_k$ | 14.1752 | 0.122583 | 34958.62 |
| *G*4 | 17.4652 | 0.001778 | 583.61 | $SHIFT_{kOPT}$ | 9.3225 | 0.158947 | 45640.85 |
| *A*1 | 6.0290 | 0.001816 | 595.67 | $BSCH_V$ ($x$ = 2) | 0.5458 | 0.025120 | 7482.71 |
| *A*2 | 9.2572 | 0.001812 | 593.95 | $BSCH_V$ ($x$ = 5) | 0.5324 | 0.052197 | 15574.12 |
| *S*1 | 19.2741 | 0.000003 | 1 | $BSCH_V$ ($x = n/10$) | 0.5295 | 0.079588 | 23690.42 |
| *S*2 | 15.2527 | 0.000006 | 2.13 | $BSCH_V$ ($x$ = 10) | 0.5349 | 0.110005 | 32796.73 |
| *S*3 | 16.6488 | 0.000007 | 2.3 | $BSCH_V$ ($x$ = 15) | 0.5359 | 0.144172 | 43052.43 |
| *TCK*1 | 15.3131 | 0.000764 | 242.55 | $BSCH_{MMA}$ ($x$ = n) | 0.5382 | 0.121981 | 36705.90 |
| *TCK*2 | 7.5253 | 0.000344 | 109.24 | $BSCH_{MMA}$ ($x = n + n/2$) | 0.5227 | 0.268500 | 78987.16 |

better estimation of the performance of all algorithms and based on the computational experiments carried out by Rad, Ruiz, and Boroojerdian (2009), Fernandez-Viagas and Framinan (2017) and Valente and Alves (2008), a total of 10 replicates for each instance are carried out and the results are averaged.

## 4.1. Performance indicators

In this section, the indicators employed to compare the different results obtained from the computational evaluation are presented. First, a comparison among the different versions of the $BSCH_{MMA}$, presented

**Table 4**
Mann–Whitney's procedure. (R indicates that the hypothesis can be rejected).

| $i$ | $H_i$ | $p$-value | Mann–Whitney | $\alpha/(k-i-1)$ | Holm's Procedure |
|---|---|---|---|---|---|
| 1 | $CH_{MMA} = A1$ | 0.000 | R | 0.0100 | R |
| 2 | $S2 = S3$ | 0.000 | R | 0.0125 | R |
| 3 | $TCK2 = TCK1$ | 0.000 | R | 0.0167 | R |
| 4 | $BSCH_V\ (x = 2) = CH_{MMA}$ | 0.000 | R | 0.0250 | R |
| 5 | $BSCH_{MMA}\ (x = n) = SHIFT_{k_{OPT}}$ | 0.000 | R | 0.0500 | R |

in Section 3.2.1, is carried out in terms of quality of the solutions and computational effort. The former is computed by means of the Average Relative Percentage Deviation (ARPD) as follows:

$$ARPD_h = \frac{\sum_{\forall s} RPD_{hs}}{S}, \quad \forall\ s = 1, \cdots, S \tag{9}$$

where $S$ is the total number of instances and $RPD$ computed as

$$RPD_{hs} = \frac{C_{hs} - C_s^*}{C_s^*} \cdot 100 \tag{10}$$

with $C_{hs}$ the total completion time obtained by heuristic $h$ ($h = 1, \cdots, H$) in instance $s$ ($s = 1, \ldots, S$) and $C_s^*$ the minimum completion time known for instance $s$. The computational effort is measured by means of the Average CPU (ACPU) time:

$$ACPU_h = \frac{\sum_{\forall s} T_{hs}}{S} \tag{11}$$

where $T_{hs}$ is the time (in seconds) required by heuristic $h$ to obtain a solution for instance $s$. Furthermore, since the $ACPU$ indicator presents some problems if it is used to compare heuristics with different stopping criteria (Fernandez-Viagas & Framinan, 2015), the Relative Percentage Indicator (labelled $RPT'$) is computed, as indicated in Eq. (12), in order to evaluate heuristics with different number of steps in their procedure.

$$RPT'_{hs} = \frac{T_{hs} - \min_{h=1,\ldots,h}\{T_{hs}\}}{\min_{h=1,\ldots,h}\{T_{hs}\}} \tag{12}$$

Additionally, a slightly different indicator, denoted $RPT$, is also used to graphically represent the results in logarithmic scale. This indicator is also employed in Fernandez-Viagas and Framinan (2015), Fernandez-Viagas and Framinan (2017) and Fernandez-Viagas, Ruiz, and Framinan (2017):

$$RPT_{hs} = \frac{T_{hs} - ACPU_s^*}{ACPU_s^*} + 1 \tag{13}$$

where $ACPU_s^*$ is the minimum CPU time obtained for instance $s$. Finally, the $ARPT$, the Average $RPT$ can be defined as follows:

$$ARPT_h = \sum_{\forall s}^{S} \frac{RPT_{hs}}{S} \tag{14}$$

### 4.2. Testbed design

In the related literature there are different testbeds for the problem proposed by Al-Anzi and Allahverdi (2006), Al-Anzi and Allahverdi (2007), Allahverdi and Al-Anzi (2009) and Al-Anzi and Allahverdi (2012). In these tests the processing times are generated in the same way, but each testbed has a different number of jobs and machines in the first stage. In the computational experiments carried out in Sections 4.3, 4.4, and 4.5, a new testbed is generated following the procedure by Al-Anzi and Allahverdi (2012). We adapt this testbed in order to consider the parameter $m_2$. Thus, the proposed testbed consists of 30 instances generated for each combination of $n$, $m_1$ and $m_2$. More specifically, the problem data are generated for $n \in \{30, 40, 50, 60, 70\}$, $m_1 \in \{2, 4, 6, 8\}$ and $m_2 \in \{2, 4, 6, 8\}$. The processing times of the jobs in the machines in the first stage are drawn from a $U[1, 100]$ distribution, while in the second stage the processing times are drawn from a $m_2 \cdot U[1, 100]$ distribution in order to balance both stages and have different scenarios regarding the relative processing times on each stage. In total, 2400 instances have been generated.

### 4.3. Experimental parameter tuning

In this section, a factorial design of experiments is performed to find the best values of the parameters of the two heuristics presented in Section 3. After some preliminary tests, it has been identified that the best values for the different parameters are in the ranges detailed below. More specifically, the next values are tested:

- Parameter $a$ for the $CH_{MMA}$ heuristic described in Section 3.1. The following levels for $a$ are tested: $a \in \{1, 2, 5, 10, 15, 20, 50, 200\}$.
- Parameters $a'$ and $b$ for the $BSCH_{MMA}$ heuristic described in Section 3.2. The following levels for each parameter are tested (in total, there are 42 combinations):
  - $a' \in \{1, 2, 5, 10, 15, 20\}$
  - $b \in \{0, 1, 2, 3, 4, 5, 6\}$

To determine the best combination of parameters, ten instances have been generated for different values of $n$, $m_1$ and $m_2$ according to the procedure described in Section 4.2. These instances are different from those used in Section 4.4 and Section 4.5 in order to avoid the overfitting of these parameters. With this testbed, after proving that the normality and homoscedasticity assumptions are not fulfilled, a non parametric Kruskal–Wallis test is performed. The results indicate that there are significant differences among the different values of parameter $a$. The best result is obtained for $a = 5$. Regarding the $BSCH_{MMA}$ heuristic, it has been assumed that $x = n$ and the so-obtained results compared. Wider ranks of parameter $a'$ have been tested but the results have not been improved. A non parametric Kruskal–Wallis test is also carried out. The results indicate that there are significant differences between parameters $a'$ and $b$ since the significance of both parameters is equal to 0.000. The best combination is obtained for $a' = 2$ and $b = 2$. In Figs. 3 and 4, the evolution of the ARPD of the heuristics according to the different experimental values of each parameter is shown. It can be seen that the minimum $ARPD$ is obtained for $a = 5$, and for $a' = 2$ and $b = 2$. These values are used for the different versions of BSCH in Section 4.4 and Section 4.5 regardless the value of $x$.

### 4.4. Comparison of the different versions of BSCH

Prior to conducting a full comparison with existing heuristics, the best variant of the beam-search based heuristics is selected. To do so, the versions of $BSCH$ presented in Section 3.2.1 have been run on the 2,400 instances generated in Section 4.2. The results are summarised in Table 2 using the indicators defined in Section 4.1. The $ARPD$ values range from 1.3196 ($BSCH_{MMA}$ ($x = 2$)) to 0.5227 ($BSCH_{MMA}$ ($x = n + n/2$)) whereas $ACPU$ values range from 0.9969 to 0.004. Results are graphically shown in Fig. 5 where the $y$-axis represents the $ARPD$ for each heuristic and the $x$-axis represents the ACPU.

In view of these results, the following conclusions can be derived:

- $BSCH_P$ ($x = 2$) with $ARPD = 0.6174$ improves the variants $BSCH_{MMA}$ ($x = 2, x = 5, x = n/10$) with $ARPD$ equal to 1.3196, 0.8064 and 0.7708 respectively, using approximately a similar computational effort.
- $BSCH_V$ ($x = 2$) with $ARPD = 0.5458$ outperforms variants $BSCH_{MMA}$ ($x = 10$), $BSCH_{ASC}$ ($x = 2$) and $BSCH_P$ ($x = 5$), with $ARPD$ equal to

**Table 5**
RPD of heuristics (I).

| $n$ | $m_1$ | $m_2$ | $A1$ | $A2$ | $S1$ | $S2$ | $S3$ | $TCK1$ | $TCK2$ | $G1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 2 | 2 | 9.36 | 9.33 | 24.48 | 11.25 | 12.84 | 19.16 | 7.36 | 16.30 |
| 30 | 2 | 4 | 5.33 | 9.44 | 19.74 | 7.93 | 12.39 | 15.15 | 5.98 | 19.17 |
| 30 | 2 | 6 | 3.85 | 7.15 | 18.30 | 6.45 | 11.07 | 13.53 | 5.77 | 18.86 |
| 30 | 2 | 8 | 3.31 | 6.94 | 15.44 | 5.32 | 10.73 | 13.37 | 6.29 | 18.86 |
| 30 | 4 | 2 | 7.34 | 10.58 | 22.35 | 18.75 | 21.02 | 16.01 | 8.84 | 16.17 |
| 30 | 4 | 4 | 5.54 | 7.86 | 17.95 | 17.39 | 17.55 | 13.70 | 6.99 | 15.98 |
| 30 | 4 | 6 | 3.39 | 7.62 | 15.93 | 14.67 | 15.38 | 12.21 | 6.44 | 15.34 |
| 30 | 4 | 8 | 1.95 | 6.39 | 14.51 | 12.74 | 13.40 | 11.33 | 5.83 | 13.48 |
| 30 | 6 | 2 | 5.61 | 9.00 | 19.89 | 18.05 | 18.27 | 13.80 | 8.67 | 13.27 |
| 30 | 6 | 4 | 4.60 | 7.97 | 16.08 | 16.23 | 16.69 | 11.84 | 8.43 | 14.94 |
| 30 | 6 | 6 | 2.80 | 6.73 | 15.47 | 14.18 | 13.91 | 11.10 | 6.31 | 13.27 |
| 30 | 6 | 8 | 2.64 | 6.18 | 13.49 | 12.68 | 12.38 | 9.86 | 5.39 | 11.64 |
| 30 | 8 | 2 | 4.03 | 8.04 | 17.86 | 17.18 | 17.04 | 11.80 | 7.94 | 12.57 |
| 30 | 8 | 4 | 3.31 | 7.36 | 14.90 | 14.87 | 15.35 | 10.30 | 8.39 | 13.85 |
| 30 | 8 | 6 | 2.72 | 6.13 | 14.98 | 13.68 | 14.58 | 9.57 | 6.13 | 12.69 |
| 30 | 8 | 8 | 2.33 | 5.91 | 13.14 | 11.94 | 12.61 | 8.87 | 6.26 | 11.57 |
| 40 | 2 | 2 | 11.38 | 10.86 | 27.11 | 12.82 | 15.34 | 22.84 | 9.06 | 19.26 |
| 40 | 2 | 4 | 9.22 | 9.29 | 24.33 | 10.82 | 12.77 | 19.74 | 7.26 | 20.49 |
| 40 | 2 | 6 | 6.13 | 7.85 | 19.13 | 7.71 | 11.36 | 16.44 | 5.95 | 20.92 |
| 40 | 2 | 8 | 4.87 | 7.45 | 19.46 | 6.72 | 10.32 | 16.60 | 5.86 | 19.14 |
| 40 | 4 | 2 | 8.46 | 11.07 | 22.56 | 22.37 | 22.90 | 17.55 | 8.98 | 17.06 |
| 40 | 4 | 4 | 4.92 | 9.47 | 20.19 | 18.49 | 19.50 | 15.50 | 8.51 | 18.81 |
| 40 | 4 | 6 | 3.58 | 8.34 | 18.17 | 15.52 | 15.49 | 13.02 | 5.61 | 15.87 |
| 40 | 4 | 8 | 3.31 | 7.01 | 16.22 | 15.26 | 14.30 | 12.72 | 6.53 | 15.03 |
| 40 | 6 | 2 | 5.92 | 10.02 | 20.21 | 18.75 | 19.99 | 15.51 | 9.43 | 15.00 |
| 40 | 6 | 4 | 3.57 | 8.71 | 17.10 | 16.59 | 17.55 | 13.15 | 7.10 | 15.11 |
| 40 | 6 | 6 | 2.86 | 7.54 | 16.58 | 15.53 | 17.83 | 11.85 | 7.30 | 15.65 |
| 40 | 6 | 8 | 2.89 | 7.58 | 14.79 | 15.21 | 15.44 | 11.56 | 6.40 | 15.14 |
| 40 | 8 | 2 | 5.22 | 10.57 | 17.48 | 17.67 | 18.60 | 13.33 | 9.45 | 14.42 |
| 40 | 8 | 4 | 4.23 | 8.73 | 17.99 | 16.60 | 17.62 | 12.62 | 7.74 | 14.61 |
| 40 | 8 | 6 | 3.28 | 7.25 | 14.88 | 13.83 | 14.39 | 11.41 | 7.27 | 13.33 |
| 40 | 8 | 8 | 3.16 | 6.88 | 14.54 | 14.53 | 13.91 | 10.41 | 6.67 | 13.36 |
| 50 | 2 | 2 | 13.65 | 11.44 | 28.81 | 14.56 | 16.52 | 23.83 | 8.69 | 21.17 |
| 50 | 2 | 4 | 9.01 | 9.32 | 22.68 | 10.80 | 14.45 | 20.56 | 6.63 | 23.75 |
| 50 | 2 | 6 | 7.78 | 8.71 | 21.55 | 9.70 | 11.89 | 18.34 | 6.06 | 23.05 |
| 50 | 2 | 8 | 6.11 | 8.32 | 21.09 | 8.14 | 12.15 | 16.47 | 6.17 | 21.82 |
| 50 | 4 | 2 | 8.52 | 11.62 | 23.51 | 20.39 | 22.21 | 19.35 | 9.24 | 17.46 |
| 50 | 4 | 4 | 6.92 | 10.35 | 21.46 | 19.96 | 20.58 | 16.82 | 7.25 | 18.59 |
| 50 | 4 | 6 | 4.99 | 9.48 | 19.46 | 17.44 | 18.16 | 14.94 | 7.15 | 18.35 |
| 50 | 4 | 8 | 4.40 | 8.23 | 18.10 | 15.77 | 17.20 | 14.46 | 6.86 | 18.02 |
| 50 | 6 | 2 | 7.28 | 10.97 | 20.47 | 19.16 | 20.66 | 16.19 | 9.43 | 16.60 |

| $n$ | $G2$ | $G3$ | $G4$ | $ECT$ | $STPT$ | $SMPT$ | $SHIFT_k$ | $SHIFT_{kOPT}$ | $SAK$ | $FAP$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 9.56 | 10.62 | 16.70 | 16.30 | 13.55 | 22.31 | 10.81 | 6.01 | 9.78 | 1.66 |
| 30 | 5.40 | 6.26 | 19.34 | 19.17 | 18.24 | 23.27 | 14.52 | 7.00 | 13.37 | 1.51 |
| 30 | 3.98 | 4.78 | 18.97 | 18.86 | 18.87 | 21.98 | 13.71 | 4.13 | 13.70 | 1.54 |
| 30 | 3.34 | 3.69 | 18.89 | 18.86 | 18.69 | 20.97 | 14.02 | 4.30 | 13.52 | 1.34 |
| 30 | 7.50 | 7.18 | 16.39 | 16.17 | 13.69 | 23.08 | 12.57 | 10.34 | 9.93 | 1.65 |
| 30 | 5.35 | 4.61 | 16.06 | 15.98 | 14.10 | 20.98 | 11.91 | 6.62 | 10.28 | 1.43 |
| 30 | 3.65 | 3.83 | 15.37 | 15.34 | 15.54 | 18.41 | 11.54 | 6.61 | 11.32 | 1.46 |

(continued on next page)

Table 5 (*continued*)

| *n* | *G2* | *G3* | *G4* | *ECT* | *STPT* | *SMPT* | $SHIFT_k$ | $SHIFT_{k_{OPT}}$ | *SAK* | *FAP* |
|---|---|---|---|---|---|---|---|---|---|---|
| 30 | 2.10 | 2.21 | 13.52 | 13.48 | 13.06 | 15.45 | 10.17 | 5.54 | 9.30 | 1.12 |
| 30 | 5.64 | 5.78 | 13.40 | 13.27 | 11.13 | 19.52 | 10.33 | 8.69 | 8.14 | 1.18 |
| 30 | 4.85 | 4.36 | 15.00 | 14.94 | 14.05 | 18.50 | 11.16 | 7.20 | 10.01 | 1.52 |
| 30 | 2.99 | 2.85 | 13.30 | 13.27 | 12.65 | 15.58 | 9.74 | 6.46 | 9.26 | 1.14 |
| 30 | 2.89 | 3.18 | 11.65 | 11.64 | 11.12 | 13.58 | 8.60 | 5.50 | 7.97 | 1.49 |
| 30 | 4.18 | 4.24 | 12.65 | 12.57 | 9.66 | 19.45 | 9.32 | 7.63 | 6.96 | 1.32 |
| 30 | 3.20 | 3.87 | 13.88 | 13.85 | 12.82 | 17.48 | 10.61 | 6.85 | 9.31 | 1.34 |
| 30 | 2.73 | 3.16 | 12.72 | 12.69 | 11.99 | 14.85 | 9.29 | 5.68 | 8.21 | 1.17 |
| 30 | 2.27 | 2.25 | 11.58 | 11.57 | 11.33 | 13.45 | 8.61 | 5.64 | 8.16 | 1.19 |
| 40 | 11.43 | 11.99 | 19.60 | 19.26 | 16.73 | 26.12 | 15.02 | 10.36 | 13.04 | 2.05 |
| 40 | 9.55 | 9.96 | 20.62 | 20.49 | 19.25 | 25.13 | 15.74 | 7.53 | 15.07 | 1.65 |
| 40 | 6.38 | 6.39 | 20.95 | 20.92 | 20.30 | 24.63 | 16.33 | 4.42 | 16.08 | 1.85 |
| 40 | 5.12 | 5.54 | 19.22 | 19.14 | 19.10 | 21.94 | 14.95 | 4.57 | 14.93 | 1.67 |
| 40 | 8.51 | 9.01 | 17.21 | 17.06 | 13.22 | 23.88 | 13.58 | 11.44 | 10.16 | 1.66 |
| 40 | 5.07 | 6.43 | 18.89 | 18.81 | 16.77 | 23.70 | 15.06 | 8.68 | 13.13 | 1.46 |
| 40 | 3.87 | 4.24 | 15.90 | 15.87 | 15.00 | 18.67 | 12.51 | 7.69 | 12.12 | 1.46 |
| 40 | 3.53 | 3.84 | 15.06 | 15.03 | 14.81 | 17.24 | 11.62 | 7.07 | 11.35 | 1.63 |
| 40 | 5.93 | 6.42 | 15.09 | 15.00 | 12.05 | 22.19 | 12.42 | 9.92 | 9.08 | 1.50 |
| 40 | 3.81 | 4.77 | 15.13 | 15.11 | 13.64 | 19.06 | 12.16 | 9.22 | 10.37 | 1.63 |
| 40 | 2.92 | 4.86 | 15.69 | 15.65 | 15.48 | 18.86 | 12.40 | 7.81 | 11.73 | 1.52 |
| 40 | 3.33 | 4.01 | 15.16 | 15.14 | 14.34 | 17.43 | 11.78 | 7.35 | 11.14 | 1.42 |
| 40 | 5.15 | 6.42 | 14.51 | 14.42 | 11.36 | 20.20 | 11.86 | 10.86 | 8.42 | 1.35 |
| 40 | 4.24 | 4.84 | 14.62 | 14.61 | 13.01 | 18.59 | 11.75 | 9.03 | 10.01 | 1.61 |
| 40 | 3.33 | 4.15 | 13.34 | 13.33 | 12.37 | 16.30 | 10.53 | 7.43 | 9.33 | 1.29 |
| 40 | 2.97 | 3.70 | 13.38 | 13.36 | 12.87 | 15.18 | 10.53 | 6.48 | 9.66 | 1.40 |
| 50 | 13.63 | 13.97 | 21.43 | 21.17 | 18.07 | 28.37 | 16.70 | 11.51 | 14.51 | 1.94 |
| 50 | 9.43 | 9.59 | 23.86 | 23.75 | 22.54 | 29.50 | 19.17 | 8.28 | 18.63 | 2.09 |
| 50 | 8.14 | 8.39 | 23.11 | 23.05 | 22.27 | 26.76 | 18.70 | 7.76 | 18.40 | 1.98 |
| 50 | 6.26 | 6.57 | 21.85 | 21.82 | 21.62 | 24.27 | 17.55 | 5.56 | 17.63 | 1.64 |
| 50 | 8.76 | 9.75 | 17.57 | 17.46 | 13.06 | 25.33 | 14.74 | 12.91 | 10.56 | 1.81 |
| 50 | 7.07 | 8.35 | 18.65 | 18.59 | 16.96 | 23.38 | 15.17 | 10.85 | 13.86 | 1.63 |
| 50 | 5.32 | 6.26 | 18.38 | 18.35 | 17.86 | 21.62 | 15.11 | 9.00 | 14.59 | 1.74 |
| 50 | 4.28 | 5.19 | 18.05 | 18.02 | 17.43 | 20.43 | 14.62 | 7.67 | 14.27 | 1.67 |
| 50 | 7.12 | 8.77 | 16.70 | 16.60 | 11.98 | 23.53 | 13.75 | 12.19 | 9.44 | 1.78 |

|  |  | $x = 2$ | $x = 5$ | $x = n/10$ | $x = 10$ | $x = 15$ | $x = n$ | $x = -n + n/2$ |
|---|---|---|---|---|---|---|---|---|
| *n* | $CH_{MMA}$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_{MMA}$ | $BSCH_{MMA}$ |
| 30 | 1.88 | 0.66 | 0.53 | 0.64 | 0.47 | 0.54 | 0.52 | 0.55 |
| 30 | 1.55 | 0.64 | 0.68 | 0.65 | 0.71 | 0.62 | 0.70 | 0.59 |
| 30 | 1.45 | 0.49 | 0.59 | 0.51 | 0.56 | 0.54 | 0.53 | 0.59 |
| 30 | 1.14 | 0.45 | 0.39 | 0.42 | 0.39 | 0.49 | 0.36 | 0.50 |
| 30 | 1.85 | 0.72 | 0.71 | 0.83 | 0.76 | 0.78 | 0.75 | 0.70 |
| 30 | 1.65 | 0.52 | 0.53 | 0.54 | 0.55 | 0.74 | 0.54 | 0.68 |
| 30 | 1.68 | 0.49 | 0.61 | 0.58 | 0.64 | 0.57 | 0.57 | 0.64 |
| 30 | 1.23 | 0.52 | 0.52 | 0.45 | 0.46 | 0.41 | 0.50 | 0.41 |
| 30 | 1.82 | 0.69 | 0.81 | 0.79 | 0.68 | 0.80 | 0.73 | 0.77 |
| 30 | 1.51 | 0.80 | 0.85 | 0.87 | 0.86 | 0.90 | 0.83 | 0.83 |
| 30 | 1.16 | 0.43 | 0.53 | 0.46 | 0.46 | 0.61 | 0.53 | 0.68 |
| 30 | 1.06 | 0.43 | 0.55 | 0.43 | 0.36 | 0.38 | 0.45 | 0.52 |
| 30 | 1.68 | 0.80 | 0.85 | 0.76 | 0.94 | 0.80 | 0.84 | 0.99 |
| 30 | 1.67 | 0.70 | 0.58 | 0.64 | 0.71 | 0.75 | 0.74 | 0.68 |

(*continued on next page*)

**Table 5** (*continued*)

| | | $x = 2$ | $x = 5$ | $x = n/10$ | $x = 10$ | $x = 15$ | $x = n$ | $x = - n + n/2$ |
|---|---|---|---|---|---|---|---|---|
| $n$ | $CH_{MMA}$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_{MMA}$ | $BSCH_{MMA}$ |
| 30 | 1.29 | 0.59 | 0.50 | 0.47 | 0.50 | 0.49 | 0.57 | 0.65 |
| 30 | 1.27 | 0.58 | 0.57 | 0.58 | 0.69 | 0.62 | 0.62 | 0.60 |
| 40 | 2.25 | 0.58 | 0.46 | 0.57 | 0.48 | 0.56 | 0.48 | 0.52 |
| 40 | 1.64 | 0.77 | 0.58 | 0.80 | 0.56 | 0.56 | 0.59 | 0.55 |
| 40 | 1.25 | 0.61 | 0.55 | 0.59 | 0.58 | 0.54 | 0.62 | 0.65 |
| 40 | 1.37 | 0.45 | 0.50 | 0.54 | 0.50 | 0.56 | 0.51 | 0.56 |
| 40 | 2.35 | 0.55 | 0.58 | 0.58 | 0.66 | 0.67 | 0.63 | 0.52 |
| 40 | 1.83 | 0.64 | 0.54 | 0.60 | 0.57 | 0.67 | 0.58 | 0.59 |
| 40 | 1.40 | 0.52 | 0.60 | 0.61 | 0.59 | 0.54 | 0.69 | 0.49 |
| 40 | 1.60 | 0.58 | 0.57 | 0.50 | 0.55 | 0.57 | 0.63 | 0.49 |
| 40 | 2.05 | 0.69 | 0.67 | 0.60 | 0.75 | 0.73 | 0.72 | 0.72 |
| 40 | 1.63 | 0.63 | 0.58 | 0.57 | 0.60 | 0.57 | 0.55 | 0.58 |
| 40 | 1.65 | 0.74 | 0.65 | 0.59 | 0.69 | 0.74 | 0.70 | 0.53 |
| 40 | 1.38 | 0.56 | 0.48 | 0.51 | 0.55 | 0.56 | 0.59 | 0.58 |
| 40 | 2.08 | 0.61 | 0.68 | 0.58 | 0.60 | 0.76 | 0.65 | 0.78 |
| 40 | 1.77 | 0.60 | 0.63 | 0.63 | 0.66 | 0.52 | 0.68 | 0.51 |
| 40 | 1.46 | 0.51 | 0.52 | 0.45 | 0.57 | 0.58 | 0.62 | 0.63 |
| 40 | 1.51 | 0.65 | 0.52 | 0.57 | 0.56 | 0.60 | 0.56 | 0.70 |
| 50 | 2.31 | 0.48 | 0.39 | 0.39 | 0.41 | 0.41 | 0.28 | 0.39 |
| 50 | 1.90 | 0.58 | 0.44 | 0.44 | 0.47 | 0.43 | 0.49 | 0.43 |
| 50 | 1.64 | 0.59 | 0.59 | 0.59 | 0.53 | 0.55 | 0.55 | 0.47 |
| 50 | 1.53 | 0.52 | 0.48 | 0.48 | 0.42 | 0.46 | 0.46 | 0.50 |
| 50 | 2.63 | 0.50 | 0.47 | 0.47 | 0.47 | 0.41 | 0.40 | 0.44 |
| 50 | 2.18 | 0.57 | 0.53 | 0.53 | 0.51 | 0.53 | 0.46 | 0.42 |
| 50 | 1.84 | 0.49 | 0.38 | 0.38 | 0.39 | 0.56 | 0.51 | 0.47 |
| 50 | 1.45 | 0.49 | 0.50 | 0.50 | 0.51 | 0.46 | 0.46 | 0.49 |
| 50 | 2.36 | 0.45 | 0.61 | 0.61 | 0.63 | 0.49 | 0.53 | 0.47 |

0.6437, 0.6145 and 0.5777 respectively.

- $BSCH_{DESC}$ ($x = 2$) with $ARPD = 0.5499$ improves variant $BSCH_{MMA}$ ($x = 15$) using the same $ACPU$.
- $BSCH_V$ ($x = 5$) with $ARPD = 0.5324$ outperforms variants $BSCH_P$ ($x = 10$) and $BSCH_{ASC}$ ($x = 5$), with $ARPD$ equal to 0.5626 and 0.5760, respectively using the same computational effort.
- $BSCH_V$ ($x = n/10$) with $ARPD = 0.5295$ outperforms variants $BSCH_{DESC}$ ($x = 5$) and $BSCH_{ASC}$ ($x = n/10$), with $ARPD$ equal to 0.5467 and 0.5815, respectively using the same computational effort.
- For $x = n$, variants $BSCH_{MMA}$ ($x = n$) and $BSCH_P$ ($x = n$) yield a similar performance, being its $ARPD$ equal to 0.5382 and 0.5348, respectively. Moreover, version $BSCH_V$ ($x = 10$) obtains a similar $ARPD = 0.5349$ with less computational effort.
- The minimum $ARPD$ is achieved by $BSCH_{MMA}$ ($x = n + n/2$), being the rest of the variants worse with respect to the quality of the solutions.
- The Pareto frontier (i.e. the efficient variants with respect to the quality of solutions and the computational effort) is formed by $BSCH_{MMA}$ ($x = 2$), $BSCH_{MMA}$ ($x = 5$), $BSCH_P$ ($x = 2$), $BSCH_V$ ($x = 2$), $BSCH_V$ ($x = 5$), $BSCH_V$ ($x = n/10$) and $BSCH_{MMA}$ ($x = n + n/2$).
- The performance of $BSCH_{MMA}$ worsens for $x = 2n$. For this value of $x$, this heuristic selects $2n$ candidates in each iteration, so there are more node candidates to be evaluated from the first iteration. Due to this poor results, it has not been considered.

To establish the statistical significance of the results, a Holm's procedure (Holm, 1979) is performed where each hypothesis is evaluated using a non-parametric Wilcoxon signed-rank test assuming a 0.95 confidence level, i.e. $\alpha = 0.05$. In Holm's test, the hypotheses are sorted in non-descending order of the $p$-values obtained in the Wilcoxon test. Each hypothesis is rejected if $p < \alpha/(k - i + 1)$ where $k$ is the total number of hypotheses. The results can be seen in Table 1, where $R$ means that the hypothesis is rejected by Wilcoxon and/or Holm's procedure. As can be seen, hypothesis $BSCH_{MMA}$ ($x = 10$) = $BSCH_{ASC}$ ($x = 2$) is the only one that cannot be rejected by Holm's procedure, but it has to be noted that the $ARPD$ achieved by the latter version, equal to 0.6145, is considerably lower than the one obtained by $BSCH_{MMA}$ ($x = 10$). In summary, it can be concluded that the variants in the Pareto frontier in Fig. 5 are efficient for the problem. However, as no variant obtains the best performance for all values of $x$, it can be also concluded that, if $x \leqslant n$ the best variant is $BSCH_V$, and if $x > n$, then $BSCH_{MMA}$ is the most efficient one.

### 4.5. Comparison of heuristics

In order to determine the performance of the proposed heuristics ($CH_{MMA}$ and the best variants of the beam-search based heuristic), these have been compared with existing heuristics for the problem, as well as with heuristics adapted from similar problems. More specifically, the heuristics used for the comparison are the following:

- Heuristics from the $DP_m \rightarrow P_m || \sum_j C_j$ problem:
  - New heuristics proposed in Section 3, i.e. heuristic $CH_{MMA}$ in Section 3.1 and the variants of the $BSCH$ proposed in Section 3.2 that are in the Pareto frontier found in 4.4: $BSCH_V$ ($x = 2$), $BSCH_V$ ($x = n/10$), $BSCH_V$ ($x = 5$), $BSCH_V$ ($x = 10$), $BSCH_V$ ($x = 15$), $BSCH_{MMA}$ ($x = n$) and $BSCH_{MMA}$ ($x = n + n/2$).
  - $SAK$ (Sung & Kim, 2008): This heuristic sorts the jobs in non decreasing order of $psum_j = \sum_{i=1,\ldots,m_1} p_{ij} + at_j$. Set $k = 1$ and $m = k + 1$, it exchanges the $k$th job and the $m$th job. If the total completion time is improved, it keeps the exchange. If not, $m = m + 1$.
  - $NSDE$ (Al-Anzi & Allahverdi, 2012): This algorithm has been coded and run, but it has been discarded due to its computational effort is far from the rest of heuristics and the quality of its solution is poor since it is specifically designed for the problem with two assembly machines.
- Heuristics adapted from the $DP_m \rightarrow 1 || \sum_j C_j$ problem. Since this problem is closely related to the one addressed in this paper, it is interesting to test whether heuristics specifically designed for the problem with one assembly machine can be adapted to the problem under consideration. These adaptations are:
  - $TCK1$ and $TCK2$ (Tozkapan et al., 2003): The original $TCK1$ constructs $m_1 + 1$ indices for each job, according to $PTF_{ij} = t_{ij}$ and $PTS_j = at_j$. So, $m_1 + 1$ sequences are obtained by sorting the jobs in non decreasing order of these indicators, and the sequence with the lowest TCT is selected. The index $PTS_j$ has been adapted to our problem so that $PTS_j = at_j/m_2$. Similarly, $TCK2$ computes three indices for each job, so three sequences are obtained by sorting the jobs in non decreasing order of these indices, and the sequence yielding the lowest TCT is selected. The indices have been adapted to our problem taking into account $m_2$, i.e.. $MPT_j = min\{p_{1j}, p_{2j}, \cdots, p_{m_1 j}, at_j/m_2\}$;$APT_j =$; and $MXPT_j =$ $\frac{1}{m1 + m_2} \sum_{i=1}^{m_1} p_{ij} + at_j/m_2$ $max\{p_{1j}, p_{2j}, \cdots, p_{m_1 j}, at_j\}$.
  - $A1$ and $A2$ (Al-Anzi & Allahverdi, 2006): These algorithms construct a sequence by iteratively appending a job at the end of a partial sequence. For algorithm $A1$, the job is chosen so that the following indicator is minimised:

$$A1_j = \max_{i=1,\ldots,m_1} \left\{ \sum_{r=1}^{j-1} p_{i[r]} + p_{ij} \right\} \tag{15}$$

  Note that this indicator does not require any adaptation to our problem. However, for algorithm $A2$ the indicator is adapted by dividing the assembly time by the number of assembly machines $m_2$, so the modified index is:

$$A2_j = \max_{i=1,\ldots,m_1} \left\{ \sum_{r=1}^{j-1} p_{i[r]} + p_{ij} \right\} + \frac{at_j}{m_2} \tag{16}$$

  - $S1$, $S2$ and $S3$ (Al-Anzi & Allahverdi, 2006): $S1$ sorts the jobs in non decreasing order of $at_j$. Heuristic $S2$ is obtained by sorting the jobs in non decresing order of $\max_{i=1,\ldots,m_1}\{p_{ij}\}$ and, finally, heuristic $S3$ orders the jobs in non decreasing order of $\max_{i=1,\ldots,m_1}\{p_{ij}\} + at_k$. As with the previous heuristics, $S1$ and $S3$ have been adapted by dividing $at_j$ by $m_2$.
  - $G1$, $G2$, $G3$ and $G4$ (Lee, 2018): Each of these heuristics constructs a sequence by inserting the job with the smallest value of one of the following indicators: $G1_j = C_{[j]} - C_2^*$;$G2_j = C1_j^* - C1_{j-1}^*$;$G3_j = C1_j^* - C_2^*$ and $G4_j = C_{[j]} - C1_j^*$. These indicators can be used for our problem in a straightforward manner.
  - FAP (Framinan & Perez-Gonzalez, 2017): This heuristic appends one by one the unscheduled jobs at the end of a partial sequence by computing an estimate of the completion times of the unscheduled jobs which takes into account which stage is more important. This estimate has been adapted considering the number of assembly machines so, if the first stage is dominant, then $FAP_l = C1_j^* + \frac{n-j+1}{n}(C_1. + \frac{at.}{m_1 + m_2})$. Otherwise, $FAP_l = \frac{at_{\omega_l}}{m_1 + m_2} + \frac{n-j+1}{n}(C_1. + \frac{at.}{m_1 + m_2})$, where $C_1.$ is the completion time in the first stage of an average artificial job composed of the unscheduled jobs, and $at.$ is the processing time of such artificial job in the second stage.
- Heuristics adapted from $DP_m \rightarrow 0 || \sum_j C_j$ problem. As mentioned before, the order scheduling problem is identical to the problem under consideration if the processing times of the second stage are zero. Therefore, it is also of interest to test how the adaptation of their best methods perform in our case. The most relevant methods

**Table 6**
RPD of heuristics (II).

| $n$ | $m_1$ | $m_2$ | $A1$ | $A2$ | $S1$ | $S2$ | $S3$ | $TCK1$ | $TCK2$ | $G1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 6 | 4 | 4.76 | 10.06 | 18.88 | 17.94 | 19.21 | 13.86 | 8.10 | 16.62 |
| 50 | 6 | 6 | 4.85 | 8.80 | 17.49 | 16.98 | 17.85 | 13.66 | 7.63 | 16.46 |
| 50 | 6 | 8 | 2.87 | 7.67 | 15.53 | 13.87 | 15.07 | 11.73 | 7.05 | 14.72 |
| 50 | 8 | 2 | 5.89 | 11.18 | 18.27 | 18.76 | 19.20 | 14.55 | 9.15 | 14.75 |
| 50 | 8 | 4 | 5.47 | 10.34 | 17.96 | 16.56 | 17.54 | 13.52 | 8.29 | 15.83 |
| 50 | 8 | 6 | 3.60 | 8.69 | 15.99 | 15.04 | 15.76 | 11.75 | 7.31 | 14.51 |
| 50 | 8 | 8 | 3.68 | 8.07 | 14.47 | 13.83 | 15.37 | 10.72 | 6.50 | 14.98 |
| 60 | 2 | 2 | 13.16 | 9.75 | 25.94 | 14.66 | 14.28 | 21.95 | 8.60 | 19.43 |
| 60 | 2 | 4 | 10.49 | 9.74 | 25.24 | 12.45 | 13.17 | 21.05 | 7.17 | 22.95 |
| 60 | 2 | 6 | 8.08 | 9.02 | 23.01 | 9.63 | 12.55 | 19.73 | 6.60 | 22.82 |
| 60 | 2 | 8 | 6.93 | 9.72 | 20.72 | 8.57 | 12.92 | 18.55 | 6.56 | 23.20 |
| 60 | 4 | 2 | 12.71 | 12.42 | 23.61 | 22.45 | 23.39 | 19.77 | 10.44 | 19.03 |
| 60 | 4 | 4 | 6.53 | 11.04 | 20.31 | 18.94 | 19.76 | 17.30 | 8.04 | 19.61 |
| 60 | 4 | 6 | 6.01 | 9.75 | 19.19 | 16.93 | 18.22 | 15.79 | 7.49 | 19.81 |
| 60 | 4 | 8 | 4.36 | 8.28 | 19.05 | 17.25 | 17.61 | 15.44 | 6.16 | 18.64 |
| 60 | 6 | 2 | 8.21 | 11.45 | 21.11 | 19.95 | 20.62 | 17.38 | 9.65 | 16.35 |
| 60 | 6 | 4 | 5.76 | 10.37 | 19.74 | 17.79 | 18.82 | 14.99 | 8.59 | 17.93 |
| 60 | 6 | 6 | 6.06 | 9.52 | 17.20 | 16.61 | 18.45 | 14.30 | 7.30 | 17.00 |
| 60 | 6 | 8 | 3.71 | 8.17 | 16.91 | 15.89 | 16.44 | 13.17 | 6.54 | 16.87 |
| 60 | 8 | 2 | 6.44 | 11.20 | 20.76 | 18.46 | 19.29 | 15.13 | 9.10 | 15.24 |
| 60 | 8 | 4 | 5.99 | 10.98 | 18.01 | 16.76 | 20.28 | 13.82 | 9.33 | 17.42 |
| 60 | 8 | 6 | 3.76 | 9.22 | 16.26 | 16.01 | 17.65 | 12.58 | 7.74 | 16.20 |
| 60 | 8 | 8 | 3.57 | 8.06 | 14.83 | 15.09 | 15.64 | 12.18 | 6.57 | 15.27 |
| 70 | 2 | 2 | 15.42 | 11.27 | 29.00 | 16.33 | 15.20 | 25.76 | 8.41 | 20.48 |
| 70 | 2 | 4 | 13.09 | 10.94 | 26.08 | 14.32 | 14.15 | 23.22 | 7.20 | 24.80 |
| 70 | 2 | 6 | 9.88 | 9.38 | 22.37 | 11.45 | 13.96 | 20.68 | 6.96 | 25.29 |
| 70 | 2 | 8 | 7.37 | 9.41 | 21.10 | 9.09 | 12.92 | 19.34 | 6.50 | 23.38 |
| 70 | 4 | 2 | 10.54 | 12.50 | 24.00 | 22.10 | 23.59 | 19.51 | 8.75 | 18.56 |
| 70 | 4 | 4 | 7.17 | 11.21 | 21.64 | 19.44 | 20.38 | 17.68 | 7.72 | 20.01 |
| 70 | 4 | 6 | 6.41 | 9.83 | 20.48 | 18.28 | 18.31 | 17.50 | 7.29 | 19.27 |
| 70 | 4 | 8 | 4.04 | 9.26 | 18.70 | 16.63 | 17.94 | 15.84 | 7.35 | 19.34 |
| 70 | 6 | 2 | 6.40 | 11.87 | 20.58 | 18.63 | 19.84 | 16.51 | 9.46 | 15.90 |
| 70 | 6 | 4 | 7.21 | 10.94 | 18.42 | 17.63 | 19.58 | 15.54 | 7.99 | 18.06 |
| 70 | 6 | 6 | 6.10 | 10.87 | 17.97 | 16.46 | 18.43 | 14.60 | 7.30 | 18.25 |
| 70 | 6 | 8 | 5.06 | 9.53 | 16.93 | 15.28 | 18.30 | 13.41 | 7.39 | 17.90 |
| 70 | 8 | 2 | 6.52 | 11.92 | 19.75 | 18.38 | 21.06 | 15.95 | 10.05 | 16.18 |
| 70 | 8 | 4 | 4.96 | 9.72 | 17.94 | 17.26 | 19.28 | 13.93 | 8.41 | 16.46 |
| 70 | 8 | 6 | 5.14 | 9.76 | 18.35 | 16.58 | 17.64 | 13.78 | 7.24 | 16.66 |
| 70 | 8 | 8 | 4.31 | 9.00 | 15.76 | 14.30 | 15.87 | 12.05 | 6.49 | 16.07 |
| Average | | | 6.73 | 10.02 | 19.73 | 16.22 | 17.58 | 16.11 | 7.81 | 18.26 |

| $n$ | $G2$ | $G3$ | $G4$ | $ECT$ | $STPT$ | $SMPT$ | $SHIFT_k$ | $SHIFT_{k_{OPT}}$ | $SAK$ | $FAP$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 5.10 | 5.49 | 16.64 | 16.62 | 15.14 | 20.95 | 13.89 | 10.43 | 12.05 | 1.49 |
| 50 | 4.70 | 4.62 | 16.48 | 16.46 | 15.71 | 19.33 | 13.50 | 9.35 | 12.51 | 1.65 |
| 50 | 3.20 | 4.08 | 14.74 | 14.72 | 14.53 | 16.94 | 11.98 | 8.64 | 11.43 | 1.55 |
| 50 | 5.93 | 7.59 | 14.78 | 14.75 | 11.00 | 20.87 | 12.49 | 11.67 | 8.94 | 1.52 |
| 50 | 5.74 | 6.65 | 15.84 | 15.83 | 14.39 | 19.54 | 13.09 | 10.98 | 11.48 | 1.31 |
| 50 | 3.63 | 4.69 | 14.51 | 14.51 | 13.90 | 17.07 | 12.16 | 9.70 | 11.26 | 1.48 |
| 50 | 3.62 | 3.80 | 14.98 | 14.98 | 14.02 | 17.14 | 12.31 | 8.71 | 11.07 | 1.37 |

(continued on next page)

**Table 6** (*continued*)

| *n* | *G2* | *G3* | *G4* | *ECT* | *STPT* | *SMPT* | $SHIFT_k$ | $SHIFT_{k_{OPT}}$ | *SAK* | *FAP* |
|---|---|---|---|---|---|---|---|---|---|---|
| 60 | 13.19 | 13.00 | 19.58 | 19.43 | 15.63 | 27.41 | 16.16 | 9.28 | 12.66 | 1.78 |
| 60 | 10.70 | 12.33 | 23.06 | 22.95 | 21.77 | 28.41 | 19.22 | 9.89 | 18.49 | 2.19 |
| 60 | 8.27 | 9.21 | 22.88 | 22.82 | 22.09 | 26.82 | 19.04 | 8.93 | 18.73 | 2.04 |
| 60 | 7.29 | 7.73 | 23.25 | 23.20 | 22.78 | 26.17 | 19.42 | 7.32 | 19.02 | 1.94 |
| 60 | 12.51 | 12.02 | 19.09 | 19.03 | 14.51 | 26.60 | 16.40 | 13.93 | 11.88 | 2.23 |
| 60 | 6.71 | 7.89 | 19.67 | 19.61 | 17.88 | 24.10 | 16.65 | 12.30 | 15.05 | 1.80 |
| 60 | 6.61 | 6.96 | 19.84 | 19.81 | 18.66 | 23.33 | 16.43 | 9.92 | 15.56 | 1.72 |
| 60 | 4.76 | 5.53 | 18.67 | 18.64 | 17.89 | 21.46 | 15.42 | 8.64 | 15.02 | 1.66 |
| 60 | 8.05 | 9.04 | 16.43 | 16.35 | 12.72 | 23.08 | 14.30 | 13.06 | 10.51 | 2.03 |
| 60 | 5.75 | 6.99 | 17.95 | 17.93 | 16.63 | 22.75 | 15.22 | 11.94 | 13.64 | 1.74 |
| 60 | 5.89 | 6.06 | 17.02 | 17.00 | 16.46 | 20.23 | 14.35 | 10.77 | 13.69 | 1.57 |
| 60 | 3.71 | 4.30 | 16.88 | 16.87 | 16.43 | 19.37 | 13.87 | 9.08 | 13.13 | 1.42 |
| 60 | 6.55 | 8.47 | 15.28 | 15.24 | 11.38 | 21.70 | 13.12 | 12.37 | 9.04 | 1.80 |
| 60 | 6.10 | 6.13 | 17.43 | 17.42 | 15.94 | 21.89 | 14.78 | 11.64 | 13.26 | 1.74 |
| 60 | 3.46 | 5.02 | 16.22 | 16.20 | 15.44 | 19.18 | 13.58 | 10.04 | 12.29 | 1.57 |
| 60 | 3.67 | 4.30 | 15.27 | 15.27 | 14.25 | 17.48 | 12.53 | 9.47 | 11.62 | 1.57 |
| 70 | 15.37 | 16.39 | 20.68 | 20.48 | 17.05 | 28.01 | 17.00 | 11.47 | 14.35 | 2.45 |
| 70 | 13.41 | 14.69 | 24.82 | 24.80 | 23.63 | 30.56 | 21.49 | 11.84 | 20.43 | 2.26 |
| 70 | 10.10 | 10.75 | 25.35 | 25.29 | 24.90 | 29.48 | 21.34 | 9.10 | 21.59 | 2.09 |
| 70 | 7.74 | 8.52 | 23.42 | 23.38 | 23.00 | 26.41 | 19.90 | 7.74 | 19.62 | 2.00 |
| 70 | 10.44 | 11.11 | 18.65 | 18.56 | 13.59 | 26.49 | 16.18 | 14.08 | 11.39 | 2.31 |
| 70 | 7.17 | 8.06 | 20.04 | 20.01 | 18.44 | 24.85 | 17.40 | 14.06 | 15.80 | 1.87 |
| 70 | 6.56 | 7.52 | 19.30 | 19.27 | 18.20 | 22.77 | 16.30 | 11.87 | 15.78 | 1.93 |
| 70 | 4.35 | 5.51 | 19.38 | 19.34 | 18.93 | 21.94 | 16.35 | 10.53 | 16.15 | 1.71 |
| 70 | 6.31 | 9.17 | 15.96 | 15.90 | 12.94 | 22.59 | 14.08 | 13.01 | 10.75 | 1.90 |
| 70 | 7.45 | 7.30 | 18.07 | 18.06 | 15.54 | 22.82 | 15.68 | 12.07 | 12.93 | 1.67 |
| 70 | 6.28 | 6.75 | 18.26 | 18.25 | 17.39 | 21.53 | 15.78 | 13.03 | 14.35 | 1.86 |
| 70 | 5.23 | 5.64 | 17.91 | 17.90 | 17.40 | 20.49 | 15.29 | 11.38 | 14.41 | 1.85 |
| 70 | 6.69 | 8.33 | 16.24 | 16.18 | 12.00 | 23.14 | 14.54 | 13.58 | 9.93 | 1.82 |
| 70 | 5.05 | 6.37 | 16.47 | 16.46 | 14.41 | 21.37 | 14.28 | 12.02 | 11.95 | 1.67 |
| 70 | 4.93 | 6.10 | 16.66 | 16.66 | 15.08 | 20.00 | 14.16 | 11.56 | 12.59 | 1.48 |
| 70 | 4.32 | 4.99 | 16.09 | 16.07 | 15.64 | 18.25 | 13.67 | 10.60 | 13.26 | 1.59 |
| Average | 6.84 | 7.67 | 18.30 | 18.26 | 16.60 | 22.63 | 15.47 | 10.92 | 13.79 | 1.79 |

|  |  | $x = 2$ | $x = 5$ | $x = n/10$ | $x = 10$ | $x = 15$ | $x = n$ | $x = -n + n/2$ |
|---|---|---|---|---|---|---|---|---|
| *n* | $CH_{MMA}$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_{MMA}$ | $BSCH_{MMA}$ |
| 50 | 1.70 | 0.47 | 0.52 | 0.52 | 0.48 | 0.49 | 0.40 | 0.42 |
| 50 | 1.61 | 0.43 | 0.50 | 0.50 | 0.63 | 0.57 | 0.54 | 0.60 |
| 50 | 1.57 | 0.52 | 0.40 | 0.40 | 0.50 | 0.46 | 0.41 | 0.42 |
| 50 | 2.14 | 0.59 | 0.49 | 0.49 | 0.49 | 0.51 | 0.51 | 0.49 |
| 50 | 1.98 | 0.62 | 0.50 | 0.50 | 0.57 | 0.52 | 0.55 | 0.50 |
| 50 | 1.82 | 0.49 | 0.41 | 0.41 | 0.42 | 0.42 | 0.44 | 0.50 |
| 50 | 1.41 | 0.54 | 0.42 | 0.42 | 0.43 | 0.52 | 0.46 | 0.48 |
| 60 | 2.41 | 0.33 | 0.27 | 0.37 | 0.31 | 0.33 | 0.48 | 0.31 |
| 60 | 2.47 | 0.57 | 0.57 | 0.57 | 0.52 | 0.51 | 0.61 | 0.39 |
| 60 | 2.09 | 0.48 | 0.58 | 0.53 | 0.60 | 0.67 | 0.52 | 0.50 |
| 60 | 2.05 | 0.70 | 0.69 | 0.63 | 0.70 | 0.57 | 0.63 | 0.54 |
| 60 | 2.76 | 0.43 | 0.33 | 0.35 | 0.42 | 0.35 | 0.34 | 0.36 |
| 60 | 2.19 | 0.53 | 0.50 | 0.51 | 0.53 | 0.47 | 0.55 | 0.48 |
| 60 | 1.79 | 0.47 | 0.48 | 0.47 | 0.51 | 0.48 | 0.47 | 0.51 |

(*continued on next page*)

**Table 6** (*continued*)

| | | $x = 2$ | $x = 5$ | $x = n/10$ | $x = 10$ | $x = 15$ | $x = n$ | $x = -n + n/2$ |
|---|---|---|---|---|---|---|---|---|
| $n$ | $CH_{MMA}$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_V$ | $BSCH_{MMA}$ | $BSCH_{MMA}$ |
| 60 | 1.72 | 0.48 | 0.56 | 0.51 | 0.47 | 0.47 | 0.53 | 0.55 |
| 60 | 2.67 | 0.49 | 0.51 | 0.54 | 0.47 | 0.37 | 0.46 | 0.47 |
| 60 | 2.11 | 0.48 | 0.49 | 0.47 | 0.50 | 0.46 | 0.48 | 0.40 |
| 60 | 1.87 | 0.53 | 0.51 | 0.53 | 0.52 | 0.47 | 0.54 | 0.41 |
| 60 | 1.59 | 0.49 | 0.45 | 0.48 | 0.47 | 0.48 | 0.48 | 0.46 |
| 60 | 2.46 | 0.58 | 0.53 | 0.46 | 0.54 | 0.51 | 0.49 | 0.45 |
| 60 | 2.20 | 0.60 | 0.56 | 0.58 | 0.56 | 0.49 | 0.58 | 0.52 |
| 60 | 1.82 | 0.47 | 0.46 | 0.37 | 0.47 | 0.45 | 0.45 | 0.36 |
| 60 | 1.68 | 0.51 | 0.50 | 0.51 | 0.44 | 0.41 | 0.49 | 0.52 |
| 70 | 2.78 | 0.42 | 0.41 | 0.46 | 0.43 | 0.40 | 0.41 | 0.35 |
| 70 | 2.31 | 0.56 | 0.61 | 0.47 | 0.45 | 0.58 | 0.52 | 0.42 |
| 70 | 1.98 | 0.61 | 0.64 | 0.61 | 0.57 | 0.66 | 0.63 | 0.60 |
| 70 | 1.94 | 0.61 | 0.59 | 0.53 | 0.49 | 0.58 | 0.61 | 0.54 |
| 70 | 2.83 | 0.44 | 0.40 | 0.44 | 0.35 | 0.38 | 0.40 | 0.34 |
| 70 | 2.31 | 0.50 | 0.44 | 0.43 | 0.46 | 0.51 | 0.55 | 0.37 |
| 70 | 2.12 | 0.53 | 0.44 | 0.52 | 0.46 | 0.45 | 0.51 | 0.46 |
| 70 | 1.91 | 0.53 | 0.55 | 0.51 | 0.44 | 0.54 | 0.48 | 0.55 |
| 70 | 2.37 | 0.43 | 0.48 | 0.38 | 0.48 | 0.41 | 0.40 | 0.43 |
| 70 | 2.37 | 0.51 | 0.47 | 0.43 | 0.47 | 0.45 | 0.43 | 0.39 |
| 70 | 2.04 | 0.51 | 0.54 | 0.60 | 0.66 | 0.66 | 0.62 | 0.48 |
| 70 | 1.78 | 0.57 | 0.71 | 0.66 | 0.63 | 0.59 | 0.62 | 0.53 |
| 70 | 2.57 | 0.40 | 0.45 | 0.53 | 0.51 | 0.41 | 0.40 | 0.47 |
| 70 | 2.01 | 0.52 | 0.51 | 0.53 | 0.49 | 0.48 | 0.54 | 0.47 |
| 70 | 1.90 | 0.45 | 0.39 | 0.40 | 0.36 | 0.35 | 0.39 | 0.47 |
| 70 | 1.62 | 0.46 | 0.44 | 0.41 | 0.44 | 0.38 | 0.43 | 0.46 |
| Average | 2.08 | 0.51 | 0.49 | 0.49 | 0.49 | 0.48 | 0.50 | 0.46 |

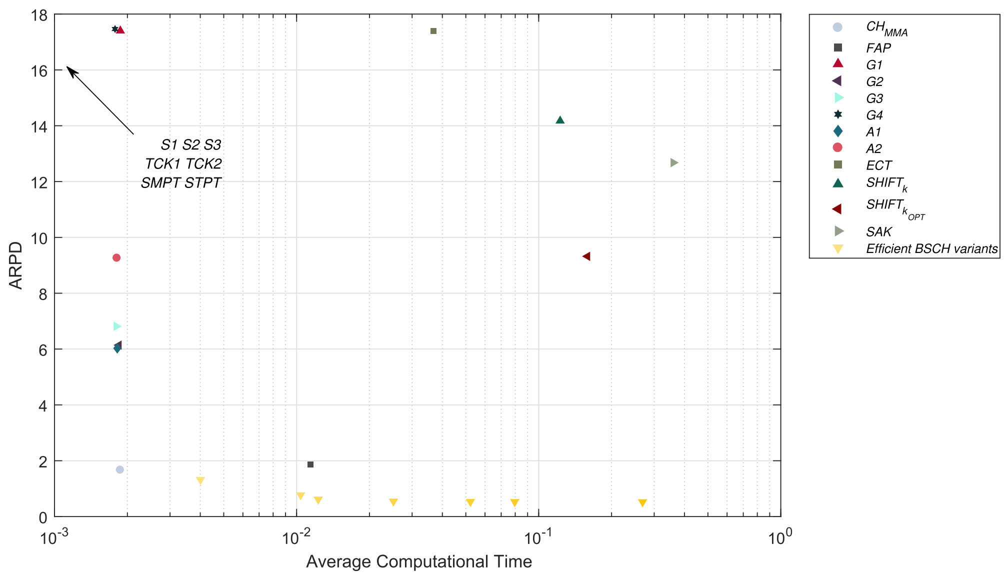


**Fig. 6.** *ARPD* versus *ACPU*. *ACPU* ($x$-axis) is shown in logarithmic scale.

for the order scheduling problem are:

- *STPT* (Sung & Yoon, 1998): A sequence is constructed sorting the jobs in ascending order of their sum of processing times on the $m_1$ machines. In our case, $m_1 + m_2$ machines are considered.
- *SMPT* (Sung & Yoon, 1998): A sequence is constructed sorting the jobs in ascending order of their maximum processing time on the $m_1$ machines. As with the previous heuristic, $m_1 + m_2$ machines are considered.
- *ECT* (Ahmadi, Bagchi, & Roemer, 2005; Leung et al., 2005): In this heuristic, the order with the earliest completion time is selected as the next to be sequenced. This heuristic does not require adaptation, as for each order, the completion time is computed according to Eq. (3).
- $SHIFT_k$ and $SHIFT_{kOPT}$ (Framinan & Perez-Gonzalez, 2017): $SHIFT_k$ obtains iteratively a partial sequence using the ECT heuristic. Then, the jobs are iteratively removed from their position and re-inserted. The procedure is repeated until the so-obtained partial sequence does not returns a lower total completion time. $SHIFT_{kOPT}$ restarts the reinsertion phase whenever a better sequence is found and repeats the process until no further improvement is found.

These heuristics have been employed to solve the instances from the testbed in Section 4.2. The *ARPD* and *ACPU* are computed according to Eqs. (9) and (11), while indicator *ARPT* is computed using Eq. (14). The detailed results of *ARPD* in terms of $n \times m_1 \times m_2$ are shown in Table 5 and 6. The average results in terms of *ARPD*, *ACPU*, and *ARPT* are shown in Table 3 and graphically in Fig. 6. Note that, in this figure, the dispatching rules are not displayed in order to have a clearer interpretation of the results. In view of the results, a number of conclusions can be noted:

- $CH_{MMA}$ ($ARPD$ = 1.8742) clearly outperforms heuristics $A1$, $A2$, $G1$, $G2$, $G3$ and $G4$ using a similar computational effort. It can be seen that $CH_{MMA}$ obtains very good results evaluating only one job at each iteration and, consequently, consuming less computational time. Furthermore, $CH_{MMA}$ obtains a similar *ARPD* than *FAP*, but our proposal requires much less CPU time.
- $S2$ and $TCK2$ with *ARPD* equal to 15.2527 and 7.5253 respectively are the best dispatching rules. Although the quality of the solution is low, these rules obtain a solution very fast.
- $BSCH_V$ ($x = 2$) with *ARPD* = 0.5458 outperforms $CH_{MMA}$, with *ARPD* equal to 1.8742, using the same computational effort, as it can be checked in Fig. 6.
- $BSCH_{MMA}$ ($x = n$) with $ARPD = 0.5382$ outperforms $SHIFT_k$, $SHIFT_{kOPT}$ and *SAK* with *ARPD* equal to 14.1752, 9.3225 and 12.6792, respectively. Moreover, it can be pointed that this version of the $BSCH_{MMA}$ obtains the best result in terms of quality of the solution, *ARPD*.
- Taking into account these results and those obtained in the previous section, the group of most efficient heuristics is formed by the dispatching rule $S2$, the existing heuristic $TCK2$ and the proposed versions of the beam search constructive heuristic: $BSCH_V$ ($x = 2$), $BSCH_V$ ($x = n/10$), $BSCH_V$ ($x = 5$), $BSCH_V$ ($x = 10$), $BSCH_V$ ($x = 15$), $BSCH_{MMA}$ ($x = n$) and ($x = 15$), $BSCH_{MMA}$ ($x = n + n/2$).

In order to check the statistical significance of these results, Holm's procedure is used as in the previous computational experience. However, each hypothesis is now analysed using a non-parametric Mann–Whitney test assuming a 95% confidence level (i.e. $\alpha = 0.05$) to establish de $p$-value of each hypothesis. The results are shown in Table 4. As it can seen, each $p$-value is 0.000, so all hypotheses can be rejected. In summary, it can be concluded that the proposed heuristics outperform the existing algorithms for the problem under consideration, as well as the adaptations of efficient algorithms for related problems.

## 5. Conclusions

In this paper we have addressed the two-stage multi-machine assembly scheduling problem with the objective of minimising the total completion time. We have presented two constructive heuristics: The first algorithm, $CH_{MMA}$, constructs a sequence by iteratively appending a job at the end of a partial sequence. The job is selected according to a problem-specific indicator that takes into account the idle time of the assembly machines in the second stage and the contribution of the job to the total completion time. Due to the good performance of this heuristic, the indicator has been embedded into a beam search based constructive heuristic, labelled $BSCH_{MMA}$, which constructs several sequences at the same time, compares them and selects the best $x$ ones. Thereby, this heuristic combines the diversification of population-based algorithms and the speed of the constructive heuristic. Furthermore, we have implemented different variants of the *BSCH*, whose main difference is the way in which the beam width ($x$) is modified in each iteration.

Using a testbed similar to Allahverdi and Al-Anzi (2012), the extensive computational experiments carried out show that the best *ARPD* are found by variants $BSCH_V$ ($\forall\ x \in \{2, n/10, 5, 10, 15, n\}$) and $BSCH_{MMA}$ ($\forall\ x \in \{n, n + n/2\}$). These variants have been compared with the $CH_{MMA}$ heuristic, and with 18 existing heuristics for the problem under consideration and heuristics adapted from related scheduling problems. The results show that the proposed heuristics outperform than the existing ones.

## Acknowledgement

The authors wish to thank the referees for their comments on the earlier versions of the manuscript. This research has been funded by the Spanish Ministry of Science and Innovation, under the project "PROMISE" with reference DPI2016-80750-P.

## References

Ahmadi, R., Bagchi, U., & Roemer, T. A. (2005). Coordinated scheduling of customer orders for quick response. *Naval Research Logistics, 52*(6), 493–512.

Al-Anzi, F. S., & Allahverdi, A. (2006). A Hybrid Tabu search heuristic for the two-stage assembly scheduling problem. *International Journal of Operations Research, 3*(2), 109–119.

Al-Anzi, F. S., & Allahverdi, A. (2006). Empirically discovering dominance relations for scheduling problems using an evolutionary algorithm. *International Journal of Production Research, 44*(22), 4701–4712.

Al-Anzi, F. S., & Allahverdi, A. (2007). A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *European Journal of Operational Research, 182*(1), 80–94.

Al-Anzi, F. S., & Allahverdi, A. (2012). Better heuristics for a two-stage multi- machine assembly scheduling problem to minimize total completion time better heuristics for a two-stage multi-machine assembly scheduling problem to minimize total completion time. *International Journal of Operations Research, 9*, 66–75.

Allahverdi, A. & Al-Anzi, F. (2012). A new heuristic for the queries scheduling problem on distributed database systems to minimize mean completion time. In *Proceedings of the 21st international conference on software engineering and data engineering*, SEDE 2012.

Allahverdi, A., & Al-Anzi, F. S. (2006). A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. *Computers and Operations Research, 33*(4), 1056–1080.

Allahverdi, A., & Al-Anzi, F. S. (2009). The two-stage assembly scheduling problem to minimize total completion time with setup times. *Computers and Operations Research, 36*(10), 2740–2747.

Allahverdi, A., & Aydilek, H. (2015). The two stage assembly flowshop scheduling problem to minimize total tardiness. *Journal of Intelligent Manufacturing, 26*(2), 225–237.

Cheng, T. E., & Wang, G. (1999). Scheduling the fabrication and assembly of components in a two-machine flowshop. *IIE Transactions, 31*(2), 135–143.

Della Croce, F., & T'kindt, V. (2003). Improving the preemptive bound for the one-machine dynamic total completion time scheduling problem. *Operations Research Letters, 31*(2), 142–148.

Erenay, F., Sabuncuoglu, I., Toptal, A., & Tiwari, M. (2010). New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs. *European Journal of Operational Research, 201*(1), 89–98.

Fattahi, P., Hosseini, S. M. H., & Jolai, F. (2013). A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 65*(5), 787–802.

Fernandez-Viagas, V., & Framinan, J. M. (2015). A new set of high-performing heuristics to minimise flowtime in permutation flowshops. *Computers and Operations Research, 53*, 68–80.

Fernandez-Viagas, V., & Framinan, J. M. (2017). A beam-search-based constructive heuristic for the PFSP to minimise total flowtime. *Computers and Operations Research, 81*, 167–177.

Fernandez-Viagas, V., Leisten, R., & Framinan, J. M. (2016). A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. *Expert Systems with Applications, 61*, 290–301.

Fernandez-Viagas, V., Ruiz, R., & Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research, 257*(3), 707–721.

Fernandez-Viagas, V., Valente, J. M. S., & Framinan, J. M. (2018). Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. *Expert Systems with Applications, 94*, 58–69.

Framinan, J. M., & Perez-Gonzalez, P. (2017). New approximate algorithms for the customer order scheduling problem with total completion time objective. *Computers and Operations Research, 78*, 181–192.

Framinan, J. M., & Perez-Gonzalez, P. (2017). The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. *Computers and Operations Research, 88*, 237–246.

Framinan, J. M., Perez-Gonzalez, P., & Fernandez-Viagas, V. (2019). Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures. *European Journal of Operational Research, 273*, 401–417.

Gonzalez, T., & Sahni, S. (1978). Flowshop and jobshop schedules: Complexity and approximation. *Operations Research, 26*(1), 36–52.

Holm, S. (1979). *Board of the Foundation of the Scandinavian Journal of Statistics, 6*(2), 65–70.

Hwang, F. J., & Lin, B. M. (2012). Two-stage assembly-type flowshop batch scheduling problem subject to a fixed job sequence. *Journal of the Operational Research Society, 63*(6), 839–845.

Jung, S., Woo, Y. B., & Kim, B. S. (2017). Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time. *Computers and Industrial Engineering, 104*, 98–113.

Koulamas, C., & Kyparisis, G. (2001). The three-stage assembly flowshop scheduling problem. *Computers and Operations Research, 28*(7), 689–704.

Lee, C.-Y., Cheng, T. C. E., & Lin, B. M. T. (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science, 39*(5), 616–625.

Lee, I. S. (2018). Minimizing total completion time in the assembly scheduling problem. *Computers and Industrial Engineering, 122*, 211–218.

Leung, J. Y. T., Li, H., & Pinedo, M. (2005). Order scheduling in an environment with dedicated resources in parallel. *Journal of Scheduling, 8*(5), 355–386.

Liao, C. J., Lee, C. H., & Lee, H. C. (2015). An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan. *Computers and Industrial Engineering, 88*(313), 317–325.

Potts, C. N., Sevast'janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., & Zwaneveld, C. M. (1995). The two-stage assembly scheduling problem: complexity and approximation. *Operations Research, 43*(2), 346–355.

Rad, S., Ruiz, R., & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *Omega, 37*(2), 331–345.

Sheikh, S., Komaki, G., & Kayvanfar, V. (2018). Multi objective two-stage assembly flow shop with release time. *Computers and Industrial Engineering, 124*, 276–292.

Shoaardebili, N., & Fattahi, P. (2015). Multi-objective meta-heuristics to solve three-stage assembly flow shop scheduling problem with machine availability constraints. *International Journal of Production Research, 53*(3), 944–968.

Sotskov, Y., Tautenhahn, T., & Werner, F. (1996). Heuristics for permutation flow shop scheduling with batch setup times. *OR Spectrum, 18*(2), 67–80.

Sung, C. S., & Kim, H. A. (2008). A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times. *International Journal of Production Economics, 113*(2), 1038–1048.

Sung, C. S., & Yoon, S. H. (1998). *Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines, 54*, 247–255.

Tozkapan, A., Kirca, Ö., & Chung, C. S. (2003). A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. *Computers and Operations Research, 30*(2), 309–320.

Valente, J. M. (2010). Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society, 61*(4), 620–631.

Valente, J. M., & Alves, R. A. (2008). Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. *Computers and Operations Research, 35*(7), 2388–2405.

Zhang, Y., Zhou, Z., & Liu, J. (2010). The production scheduling problem in a multi-page invoice printing system. *Computers and Operations Research, 37*(10), 1814–1821.