Computers & Operations Research 40 (2013) 639–653

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Two-machine flowshop scheduling with flexible operations and controllable processing times

Zeynep Uruk [a], Hakan Gultekin [b,*], M. Selim Akturk [a]

[a] Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey
[b] Department of Industrial Engineering, TOBB University of Economics and Technology, 06560 Ankara, Turkey

ARTICLE INFO

Available online 11 September 2012

Keywords:
Flexible manufacturing system
Controllable processing times
Flowshop
Scheduling

ABSTRACT

We consider a two-machine flowshop scheduling problem with identical jobs. Each of these jobs has three operations, where the first operation must be performed on the first machine, the second operation must be performed on the second machine, and the third operation (named as flexible operation) can be performed on either machine but cannot be preempted. Highly flexible CNC machines are capable of performing different operations. Furthermore, the processing times on these machines can be changed easily in albeit of higher manufacturing cost by adjusting the machining parameters like the speed and/or feed rate of the machine. The overall problem is to determine the assignment of the flexible operations to the machines and processing times for each operation to minimize the total manufacturing cost and makespan simultaneously. For such a bicriteria problem, there is no unique optimum but a set of nondominated solutions. Using ϵ-constraint approach, the problem could be transformed to be minimizing total manufacturing cost for a given upper limit on the makespan. The resulting single criterion problem can be reformulated as a mixed integer nonlinear problem with a set of linear constraints. We use this formulation to optimally solve small instances of the problem while a heuristic procedure is constructed to solve larger instances in a reasonable time.

1. Introduction

In this paper, we study the problem of scheduling n identical jobs each of which has three operations to be performed on two machines placed in series. One of the operations can only be performed on the first, the other one by the second machine. The third operation is flexible meaning that it can be performed by either one of the machines. Besides such flexible operations, we also consider the controllability of the processing times of each operation on these machines. In the scheduling literature, there are a number of studies considering flexible operations and controllable processing times separately. However, this is the first study that considers both of these simultaneously through a bicriteria objective.

In most of the deterministic scheduling problems in the literature, job processing times are considered as constant parameters. However, various real-life systems allow us to control the processing times by allocating extra resources, such as energy, money, or additional manpower. Under controllable processing times setting, the processing times of the jobs are not fixed in advance but chosen from a given interval. The processes on the CNC machines are well known examples of how the processing times can be controlled. By adjusting the speed and/or feed rate, the processing times on these machines can easily be controlled. Although reducing the processing times may lead an increase in the throughput rate, it incurs extra costs as well. Controllability of processing times may also provide additional flexibility in finding solutions to the scheduling problem, which in turn can improve the overall performance of the production system. Therefore, in such systems we need to consider the trade-off between job scheduling and resource allocation decisions carefully to achieve the best scheduling performance.

Study of the controllable processing times in scheduling was initialized by Vickson [16]. He drew attention to the problems of least cost scheduling on a single machine in which processing times of jobs were controllable. Nowicki and Zdrzalka [9] worked on two-machine flowshop scheduling problems, for which Janiak [4] showed that the problem of minimizing makespan is NP-hard for a two-machine flowshop with linear compression costs. Karabati and Kouvelis [6] discussed simultaneous scheduling and optimal processing time decision problem for a multi-product, deterministic flow line operated under a cyclic scheduling approach. Yedidsion et al. [18] considered a bicriteria scheduling problem of controllable assignment costs and total resource consumption. Wang and Wang [17] studied a single machine scheduling problem to minimize total convex resource consumption cost for a given upper bound on the total

* Corresponding author.
E-mail address: hgultekin@etu.edu.tr (H. Gultekin).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.09.001

weighted flow time. Shabtay et al. [13] studied a no-wait two-machine flowshop scheduling problem with controllable job-processing times under the objective of determining the sequence of the jobs and the resource allocation for each job on both machines in order to minimize the makespan. Gultekin et al. [2] considered a cyclic scheduling environment through a flowshop type setting in which identical parts were processed. The parts were processed with two identical CNC machines and transportation of the parts between the machines was performed by a robot. Both the allocations of the operations to the two machines and the processing time of an operation on a machine were assumed to be controllable. Shabtay et al. [12] proposed a bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the resource consumption cost in a two-machine flowshop environment. Shabtay and Steiner [14] provided a survey of the results in the field of scheduling with controllable processing times.

There are several studies on decision rules for the assignment of the flexible operations in fixed processing time production environment. Gupta et al. [3] studied a two-machine flowshop processing nonidentical jobs that the buffer has infinite capacity. Each job had three operations, one of which was a flexible operation. The assignment of the flexible operations to the machines for each job was determined under the objective of maximizing the throughput rate. They showed that the problem is NP-Hard and developed a 3/2-approximation algorithm and a Polynomial Time Approximation Scheme (PTAS). Crama and Gultekin [1] considered the same problem for identical parts, under different assumptions regarding the number of jobs to be processed and the capacity of the buffer in between the machines. For each problem, alternative polynomial time solution procedures are developed. Ruiz-Torres et al. [11] studied a flowshop scheduling problem with operation and resource flexibility to minimize the number of tardy jobs.

Our study is the first one that considers both assignment of the flexible operations and the controllability of processing times at the same time through a bicriteria objective. We assume the processing times to be controllable with nonlinear manufacturing cost functions. As a consequence of the controllability of the processing times and the dynamic assignment of the flexible operations from one part to the other, although the jobs are assumed to be identical they may have different processing times on the machines. Consequently, they are identical in the sense that, they all require the same set of operations. The problem is to determine the assignment of flexible operations to one of the machines along with the processing time of each operation under the bicriteria objective of minimizing the total manufacturing cost and makespan.

The rest of the paper is organized as follows. In the next section, we state the problem definition and formulate it as a nonlinear mixed integer problem to determine a set of efficient discrete points of makespan and manufacturing cost objectives. In Section 3, we demonstrate some basic properties for the problem which will be used in the development of the algorithm that will be discussed in Section 4. We perform a computational study in Section 5 to test the performance of our proposed algorithm by comparing it with an exact approach. Section 6 is devoted to concluding remarks and possible future research directions.

2. Problem definition and modeling

There are n identical jobs requiring three operations to be performed by the two machines placed in series. There is always space for the new parts in the buffer space between the machines and preemption is not allowed. All jobs are first processed by the first machine and then by the second machine. The first (second) operation can only be performed by the first (second) machine. The third operation can be performed by either one of the machines. Due to the flowshop nature of the problem, the third operation must be performed after the first or before the second operation as in Gupta et al. [3] and Crama and Gultekin [1]. The assignment of the flexible operation to one of the machines for each job is a decision that should be made.

Furthermore, the processing times are not fixed predefined parameters, but they are controllable and can take any value in between a given lower and an upper bounds. For job j, the processing times of the fixed operations on the first and the second machines are denoted by f_j^1 and f_j^2, respectively and the processing time of the flexible operation is denoted by s_j. These denote the actual processing times on the machines. The parts are identical in the sense that their processing time functions are job independent. However, actual processing times of the parts on the machines may differ from one job to another. The second decision is to determine the values of these processing times.

The manufacturing cost of an operation for the CNC machines can be expressed as the sum of the operating and the tooling costs. Operating cost of a machine is the cost of running this machine. Tooling cost can be calculated by the cost of the tools used times tool usage rate of the operation. Kayan and Akturk [7] showed that manufacturing cost of a turning operation can be expressed as a nonlinear function of its processing time. Although we consider the manufacturing cost incurred for a CNC machine, our analysis is valid for any convex differentiable manufacturing cost function.

We present the notation used throughout the paper below. Note that, since the jobs are identical, the index j denotes the job in the jth position.

Decision variables

f_j^i	processing time of the preassigned operation of job j on machine i, $i=1,2$ and $j=1,..,n$
s_j	processing time of the flexible operation of job j on the assigned machine
x_j	decision variable that controls if flexible operation of job j is assigned to machine 1 or not
$T_{j,i}$	starting time of the jth job on machine i
$C_{j,i}$	completion time of the jth job on machine i

Parameters

J	set of jobs to be processed
n	number of jobs to be processed, $\|J\|=n$
O	operating cost coefficient of machines ($/time unit)
$F^i(f_j^i)$	manufacturing cost function incurred by job j on machine i
$S(s_j)$	manufacturing cost function incurred by the flexible operation of job j on the assigned machine
f_l^i, f_u^i	processing time lower and upper bounds of the preassigned operations on machine i, respectively
s_l, s_u	processing time lower and upper bounds for the flexible operations, respectively
b	tooling cost exponent (note that, $b<0$)
A^1, A^2, A^s	tooling cost multipliers for the 1st, 2nd, and the flexible operations, respectively

Having these notations, the manufacturing cost functions for the preassigned and flexible operations can now be written as follows:

$$F^i(f_j^i) = O \cdot f_j^i + A^i \cdot (f_j^i)^b \quad \text{for } i=1,2 \text{ and } j=1,\ldots,n \tag{1}$$

$$S(s_j) = O \cdot s_j + A^s \cdot (s_j)^b \quad \text{for } j = 1, \ldots, n \tag{2}$$

Note that, due to physical constraints of the manufacturing properties of job j and the maximum applicable power of CNC machines the processing times cannot be reduced indefinitely. Therefore, we use the lower bounds, f_l^1, f_l^2, and s_l, for operation process times.

We can formulate the bicriteria problem as a mixed integer nonlinear program as follows:

$$\text{Min} \quad Z_1 = \left(\sum_{j=1}^{n} \sum_{i=1}^{2} f_j^i + \sum_{j=1}^{n} s_j \right) \cdot O + \sum_{i=1}^{2} \sum_{j=1}^{n} (f_j^i)^b \cdot A^i + \sum_{j=1}^{n} (s_j)^b \cdot A^s \tag{3}$$

$$\text{Min} \quad Z_2 = T_{n,2} + f_n^2 + s_n \cdot (1 - x_n) \tag{4}$$

$$\text{s.t.} \quad T_{j,1} \geq T_{j-1,1} + f_{j-1}^1 + s_{j-1} \cdot x_{j-1}, \quad j \geq 2 \tag{5}$$

$$T_{j,2} \geq T_{j-1,2} + f_{j-1}^2 + s_{j-1} \cdot (1 - x_{j-1}), \quad j \geq 2 \tag{6}$$

$$T_{j,2} \geq T_{j,1} + f_j^1 + s_j \cdot x_j \quad \forall j \tag{7}$$

$$T_{1,1} \geq 0 \tag{8}$$

$$f_j^i \geq f_l^i \quad \forall i \text{ and } \forall j \tag{9}$$

$$s_j \geq s_l \quad \forall j \tag{10}$$

$$x_j \in \{0,1\} \quad \forall j \tag{11}$$

Eq. (3) is the first objective function, which minimizes the total manufacturing cost. Eq. (4) is the second objective function which minimizes makespan. Constraints (5) and (6) express the condition that the jth job can start on the first (resp., second) machine only after the previous job is completed on this machine. Constraint (7) states that the processing of a job on the second machine can be started only after the processing of this job is completed on the first machine. Constraint (8) is the nonnegativity constraint of the variable $T_{1,1}$. Lower bounds of the processing times of the first, second, and flexible operations are represented by the Constraints (9) and (10), respectively.

Since the formulation has two challenging objectives, there is no unique optimal solution but an infinite set of nondominated (efficient) solutions exist. T'kindt and Billaut [15] defined that a point (Z_1^b, Z_2^b) is said to be efficient with respect to cost and makespan criteria if there does not exist another point (Z_1^d, Z_2^d) such that $Z_1^d \leq Z_1^b$ and $Z_2^d \leq Z_2^b$ with at least one holding as a strict inequality. As discussed by these authors, one of the methods used in the literature for generating nondominated solutions for such bicriteria problems is the so-called ϵ-constraint approach. This method represents one of the objectives as a constraint with an auxiliary upper bound and optimizes over the second objective. By searching over different values for the upper bound one can generate a set of discrete nondominated points. We will make use of this approach to generate nondominated solutions for our bicriteria problem. Manufacturing cost objective is a convex nonlinear function which cannot be linearized. On the other hand, makespan objective is a nonlinear function which can be linearized with a reformulation. We use the makespan as a constraint and optimize over the manufacturing cost objective in order not to have the nonlinearity in the constraint set. Therefore, our problem turns out to be minimizing total manufacturing cost for a given upper limit, ϵ, on the makespan.

Model 1 : Min Z_1

$$\text{s.t.} \quad Z_2 \leq \epsilon$$
$$\text{Constraints (5)}-\text{(11)} \tag{12}$$

Constraints (5), (6), (7), and (12) of Model 1 includes the multiplication of two variables. After replacing $s_j \cdot x_j$ with y_j^1 and $s_j \cdot (1-x_j)$ with y_j^2, these constraints can be linearized which yields the following model with a nonlinear objective function but a set of linear constraints. Here y_j^i represents the processing time for flexible operation of job j on machine i and M is a large number.

Model 2 : Min Z_1

$$\text{s.t.} \quad T_{n,2} + f_n^2 + y_n^2 \leq \epsilon \tag{13}$$

$$T_{j,1} \geq T_{j-1,1} + f_{j-1}^1 + y_{j-1}^1, \quad j \geq 2 \tag{14}$$

$$T_{j,2} \geq T_{j-1,2} + f_{j-1}^2 + y_{j-1}^2, \quad j \geq 2 \tag{15}$$

$$T_{j,2} \geq T_{j,1} + f_j^1 + y_j^1 \quad \forall j \tag{16}$$

$$s_j - M \cdot (1 - x_j) \leq y_j^1 \leq s_j + M \cdot (1 - x_j) \quad \forall j \tag{17}$$

$$-M \cdot x_j \leq y_j^1 \leq M \cdot x_j \quad \forall j \tag{18}$$

$$y_j^2 = s_j - y_j^1 \quad \forall j \tag{19}$$

$$\text{Constraints (8)}-\text{(11)} \tag{20}$$

2.1. Characteristics of the problem

The cost functions given in Eqs. (1) and (2) are strictly convex and have unique minimizers for $f_j^i, s_j > 0$. Kayan and Akturk [7] showed that a processing time value greater than the minimizer of the cost function is inferior both in terms of the manufacturing cost and any regular scheduling measure. Therefore, the optimal processing time values will never exceed the minimizers of these functions which are the natural upper bounds of processing times. These are denoted by f_u^i and s_u. Since the cost functions are convex and differentiable, these minimizers can be determined using the derivatives. Then, we have $f_u^1 = \sqrt[b-1]{-O/(A^1 \cdot b)}$, $f_u^2 = \sqrt[b-1]{-O/(A^2 \cdot b)}$, and $s_u = \sqrt[b-1]{-O/(A^s \cdot b)}$. Note that, the manufacturing cost function is a monotonically decreasing function for $f_j^i \leq f_u^i$ and $s_j \leq s_u$.

Crama and Gultekin [1] showed that the optimal assignment of flexible operations in fixed processing times case can be found in polynomial time. Using the same idea behind Johnson's algorithm [5], for the first $n-r$ jobs the flexible operations are assigned to the second machine and for the remaining ones they are assigned to the first machine. Here, r can be calculated using the following formula:

$$r = \frac{(n-1) \cdot (f^2 + s - f^1) + s}{2 \cdot s} \tag{21}$$

By definition, r must be an integer. If this equation produces a noninteger value, then the optimal makespan, C_{max}, is found using either the largest integer smaller than r, $\lfloor r \rfloor$, or the smallest integer larger than r, $\lceil r \rceil$, and the following formula holds:

$$C_{max} = \min\{(f^1 + n \cdot f^2 + (n - \lfloor r \rfloor) \cdot s), (n \cdot f^1 + f^2 + \lceil r \rceil \cdot s)\} \tag{22}$$

Fig. 1 represents an efficient frontier of makespan and total manufacturing cost objectives. In this figure, Z_1 denotes the manufacturing cost and Z_2 denotes the makespan value.

The processing time upper bounds are preferred in terms of manufacturing cost objective because processing the jobs at their upper bounds allows to achieve minimum manufacturing cost. At point C in Fig. 1, Z_1 is at the minimum value (Z_1^{min}) while Z_2 is at the maximum value (Z_2^{max}). This point is reached when the processing times are set to their upper bounds.

When we focus on any regular scheduling measure, the smaller the processing times, the better the objective function value. When the processing times are set to their lower bounds, we get point A in Fig. 1. However, if the calculated r value is not an integer, one of the machines will be idle which is unavoidable when the processing times are assumed to be fixed. This may incur extra cost. However controllability allows us to prevent idleness by increasing some of the processing times of jobs and changing the assignment of the flexible operations. Increasing the processing times to cover the idle time reduces the manufacturing cost without increasing the makespan. Therefore, a schedule can be obtained with the same makespan value but at a lower cost. This idea will be highlighted via examples in the next section. In such a situation, old solution A becomes a dominated point. The new point is a nondominated point, which is represented by B in Fig. 1. At point B, Z_1 has its maximum value (Z_1^{max}) and Z_2 has its minimum (Z_2^{min}).

2.2. Numerical examples

In this section, we present two examples to demonstrate the benefits of controllable processing times over fixed processing times case.

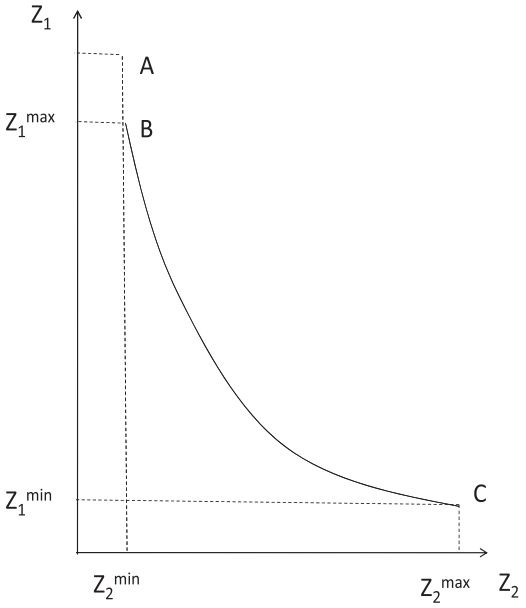

Fig. 1. Efficient frontier of makespan and total manufacturing cost objectives.

Example 1. Let us consider two cases, one of which assumes fixed processing times and the other has controllable processing times.

*Case*1: Fixed processing time case.

Let $n=5$, the processing times be $f_j^1=1.2$, $f_j^2=2$, and $s_j=1.8$ $\forall j$, and the operating cost of machines, tooling cost multiplier, and tooling cost exponent be $O=4$, $A=8$, and $b=-2$, respectively. Using the solution procedure developed by Crama and Gultekin [1], the optimal makespan value is found to be 14.8 time units. The Gantt chart of the optimal solution is depicted in Fig. 2a, in which the flexible operations of jobs 3, 4, and 5 are assigned to the first and the remaining ones are assigned to the second machine. With given parameters, the associated total manufacturing cost of this solution is 62.62.

*Case*2: Controllable processing time case.

Let us consider the same parameters as in Case 1 with the addition that the processing times now given as ranges $1.2\le f_j^1\le 4.7$, $2\le f_j^2\le 2.8$, and $1.8\le s_j\le 5.6$, $\forall j$. Let us use the optima of Case 1 as the upper limit in Constraint (12), i.e., $\epsilon=14.8$.

The problem is solved using BARON MINLP solver of GAMS and because of the convex nature of the problem the solver guarantees an optimal solution. The solution is found to be $f_1^1=1.2$, $f_j^1=1.55$ for $2\le j\le 5$, $f_j^2=2$, $\forall j$, and $s_j=1.8$, $\forall j$. In Fig. 2b, the optimal solution of the problem with cost 54.42 and makespan 14.8 time units is depicted.

As can be seen from Fig. 2a and b, while there is an idle time in the schedule of Case 1, there is no idle time in Case 2. By means of controllability of processing times, f_j^1 for jobs $2\le j\le 5$ are increased, which resulted in a reduction in the manufacturing cost by 15.1%. On the other hand, the makespan value is the same for both solutions, and hence the old solution is dominated.

In this example, the assignment of the flexible operations remained the same but processing times are changed in the flexible system. As shown in the next example in some cases, changing the assignments in addition to the processing times could provide better results.

Example 2. *Case*1: Fixed processing time case.

Let us use the same parameters as in Example 1 except the processing times $f_j^1=1.8$, $f_j^2=2.4$, and $s_j=4.5$, $\forall j$. Fig. 3a depicts the optimal solution of the problem with makespan 24.9 time units and the corresponding manufacturing cost is 43.01.

*Case*2: Controllable processing time case.

Let us use the same parameters as in Case 2 of Example 1. Using the makespan value attained in Case 1 of this example as the upper limit in Constraint (12), we get $f_1^1=2.77$, $f_j^1=3.17$ for $2\le j\le 5$, $f_j^2=2.77$, $\forall j$, $s_j=2.77$ for $j\le 3$, and $s_j=3.17$ for $j\ge 4$. Fig. 3b depicts the optimal solution of the problem with cost 36.14 and makespan 24.9 time units.

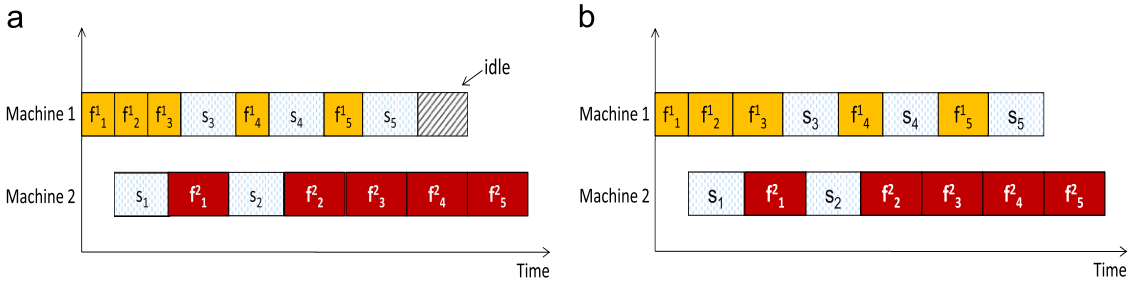

Fig. 2. Gantt charts for the two cases in Example 1. (a) Case 1. (b) Case 2.

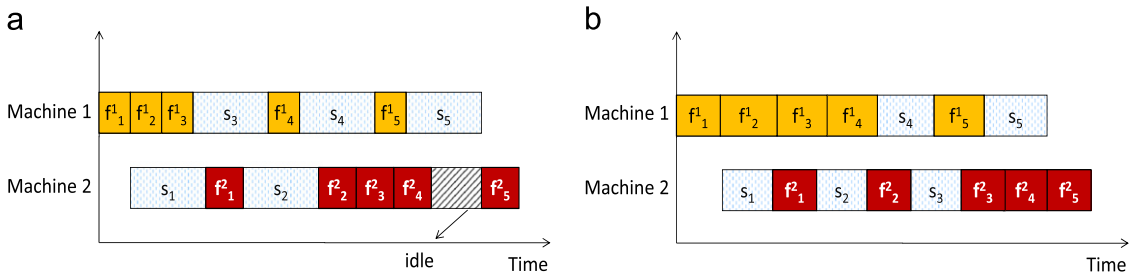

Fig. 3. Optimal schedule of Example 2. (a) Case 1. (b) Case 2.

As can be seen from Fig. 3a and b, while there is an idle time in the schedule of Case 1, there is no idle time in the schedule of Case 2. In the optimal schedule of Case 1, the flexible operations of jobs 3, 4, and 5 are assigned to the first machine and the other two to the second machine. However, in the optimal schedule of Case 2, the flexible operations of jobs 4 and 5 are processed on the first machine and the other three on the second machine. The processing times of operations are also different than the fixed processing time case. While f_j^i, $\forall i,j$ are increased, s_j, $\forall j$ are decreased. The controllability of processing times decreased the manufacturing cost by 19.0% in this case. At the end, a new schedule is obtained with the same makespan at a lower cost.

As highlighted in these examples, the solution procedure proposed by Crama and Gultekin [1], which determines the optimal solutions for fixed processing times case, may not be optimal for controllable processing times case. Moreover, although the jobs are assumed to be identical in this study, they may have different processing times.

3. Theoretical results

In this section, we demonstrate some optimality properties for the problem which will be used in the development of the algorithm that will be presented in the next section. The first lemma considers the property of the second objective function represented as a constraint in Model 1.

Lemma 1. *In an optimal solution to the problem, either* $Z_2=\epsilon$ *or* $f_j^i=f_u^i$ $\forall j,i$.

Proof. Let $Z_1^\star=\sum_{j=1}^{n}\sum_{i=1}^{2}(F_j^i(f_j^{i^*})+S_j(s_j^\star))$ be the optimal objective function value with optimal processing time vectors $f^\star$ and $s^\star$. Assume to the contradiction that $Z_2=T_{n,2}+f_n^2+s_n\cdot(1-x_n)<\epsilon$ and there exists $k\in J$, such that $f_k^i<f_u^i$. Consider another solution with $\hat{f}_j^i=f_j^{i^*}$, $\forall j\neq k$ and $\hat{f}_k^i=f_k^{i^*}+\beta$, $0<\beta\leq\min\{f_u^i-f_k^{i^*},\epsilon-(T_{n,2}+f_n^2+s_n\cdot(1-x_n))\}$. If all processing times are at their upper bounds or if $Z_2=\epsilon$, there is no such β. Otherwise, this new solution is still feasible for Model 1. Processing times for all jobs except k is identical with the previous solution and $\hat{f}_k^i>f_k^{i^*}$. Since the cost function is decreasing with respect to the processing times for $f_j^i\leq f_u^i$, we have $\hat{Z}_1<Z_1^\star$. This contradicts with $f^\star$ being the optimal solution. □

As a consequence of this lemma, in an optimal schedule we know that either all processing times are set to their upper bounds or makespan value (C_{max}) is equal to the upper bound ϵ.

In any schedule, both of the machines are initially idle. Then, while the first job is being processed on the first machine, the second machine is still idle waiting for the job during the interval $[0,T_{1,2}]$. This idle time on the second machine cannot be avoided and its length is equal to f_1^1. Similarly, some idle time of length f_n^2 cannot be avoided on the first machine while the second machine processes the last job. All other idle times except these are named as unforced idle times and can be eliminated by reassignment of the flexible operations or by changing the processing time values. Constraint (7) may yield unforced idle times on machine 2 during the time interval $[T_{1,2},T_{n,2}]$. On the other hand, since all parts are ready in front of machine 1 and there is always space for a part in the buffer in between the machines, no unforced idle time can occur on machine 1 during the time interval $[T_{1,1},C_{n,1}]$. However, if after the last part is completed on machine 1, machine 2 is still busy processing the previous parts, some unforced idle time can occur on machine 1. As shown in the numerical examples by eliminating unforced idle times the manufacturing cost can be reduced without increasing the makespan. The following lemma characterizes the occurrence of such idle times in the optimal schedule.

Lemma 2. *In an optimal solution to the problem, the following conditions hold.*

1. *Either* $C_{n-1,2}\leq C_{n,1}$ *or* $f_j^1=f_u^1$ $\forall j\in J$.
2. *Either* $C_{k,1}\leq C_{k-1,2}$ *or* $f_j^2=f_u^2$ $\forall k\in J$ *and* $\forall j=1,2,\ldots,k-1$.

Proof. Let $Z_1^\star$, $f^\star$, and $s^\star$ denote the optimal objective function value, optimal processing time vector for fixed and flexible operations, respectively.

1. Assume to the contradiction that $C_{n-1,2}>C_{n,1}$ and there exists k such that $f_k^1<f_u^1$. Now, consider another solution with $\hat{f}_j^1=f_j^{1^*}$ $\forall j\neq k$ and $\hat{f}_k^1=f_k^{1^*}+\min\{f_u^1-f_k^{1^*},C_{n-1,2}-C_{n,1}\}$. Any solution formed in this way is still feasible for Model 1 and $\hat{f}_j^1\geq f_j^{1^*}$, $\forall j$. Since the cost function is decreasing with respect to the processing times, $\hat{Z}_1<Z_1^\star$. However, this contradicts with $f^\star$ being the optimal solution.
2. Assume to the contradiction that there exists k such that $C_{k,1}>C_{k-1,2}$ and there exists h, $1\leq h\leq(k-1)$, such that $f_h^2<f_u^2$. Among multiple occurrences of such k, select the smallest one. Now, consider another solution with $\hat{f}_j^2=f_j^{2^*}$, $\forall j\neq h$ and $\hat{f}_h^2=f_h^{2^*}+\min\{f_u^2-f_h^{2^*},C_{k,1}-C_{k-1,2}\}$. This new solution is still feasible for Model 1. Since $\hat{f}_j^2>f_j^{2^*}$, $\forall j$ we have $\hat{Z}_1<Z_1^\star$. However, this contradicts with $f^\star$ being the optimal solution. □

Lemma 2 indicates that in an optimal schedule the unforced idleness of machines must be prevented till the processing time variables reach to their upper bounds. If the first and the second statements of the lemma are combined, then either $C_{n-1,2}=C_{n,1}$ or $f_j^1=f_u^1$ $\forall j$, or $f_j^2=f_u^2$ $\forall j$.

Lemma 3. *There exists an optimal schedule in which* $x_j=0$ *for* $j=1,2,\ldots,(n-r)$ *and* $x_j=1$ *for* $j=(n-r+1),(n-r+2),\ldots,n$.

This lemma can be proved in a similar way as in Crama and Gultekin [1] and is left to the reader. As a result of this lemma we know that there exists an optimal schedule in which the flexible operations are assigned to the second machine for the first $(n-r)$

parts and to the first machine for the remaining r parts. This result will be useful in proving the following lemmas. The following property, resulting from the convexity of the cost function, will also be used in these proves. Although it is proved for $F^i(f_j^i)$ function, it is also valid for the flexible operations, $S(s_j)$.

Property 1. *If $f_j^i < f_k^i$ then $F^i(f_j^i+\delta)+F^i(f_k^i-\delta) \le F^i(f_j^i)+F^i(f_k^i)$, for $0 \le \delta \le (f_k^i-f_j^i)/2$, $1 \le j,k \le n$, and $i=1, 2$.*

Proof. For $0 \le \delta \le (f_k^i-f_j^i)/2$, we can write $f_j^i+\delta = \alpha f_j^i+(1-\alpha)f_k^i$, and $f_k^i-\delta = (1-\alpha)f_j^i+\alpha f_k^i$, $\alpha \in [0,0.5]$. Since F^i is a convex function, $F^i(\alpha a+(1-\alpha)b) \le \alpha F^i(a)+(1-\alpha)F^i(b)$, for $\alpha \in [0,1]$. Using this property, we have the following:

$$\begin{aligned} F^i(f_j^i+\delta)+F^i(f_k^i-\delta) &= F^i(\alpha f_j^i+(1-\alpha)f_k^i)+F^i((1-\alpha)f_j^i+\alpha f_k^i) \\ &\le \alpha F^i(f_j^i)+(1-\alpha)F^i(f_k^i)+(1-\alpha)F^i(f_j^i)+\alpha F^i(f_k^i) \\ &= F^i(f_j^i)+F^i(f_k^i) \qquad \square \end{aligned}$$

As an implication of this lemma we can conclude that, in order to minimize the cost, the processing times of the same type (f_j^i) should be equal to each other. However, in order to prove this statement one must consider the feasibility of the schedule which is done in the following lemma.

Note that, when the order of the machines is reversed and when the first and the second machines are switched we get the "reversed" problem which has the identical objective function value as the original problem [8]. The optimal solution for one of these problems is the symmetric of the other one. Therefore, any solution proved for one of the machines can be adopted to the other one. Using this property, we prove the following for the second machine and the result for the first machine is presented as Corollary 1 without proof.

Lemma 4. *In the optimal schedule $f_j^2 = f_k^2$ for $1 \le j,k \le n-1$ and $f_n^2 \le f_j^2$, for $1 \le j \le n-1$.*

Proof. Let us first prove the second part of the lemma. Assume to the contradiction that $f_n^2 > f_k^2$ for at least one index $k \le n-1$. Then, we can construct a new schedule as $\hat{f}_n^2 = f_n^2-\delta$ and $\hat{f}_k^2 = f_k^2+\delta$. As a consequence of this change the completion times $\hat{C}_{j,2} \ge C_{j,2}$ for $j=k,k+1,\ldots,n-1$ and $\hat{C}_{n,2} = C_{n,2}$. Therefore, without making any other change, the final schedule is still feasible and from Property 1 its cost is not increased. Therefore, the new schedule is also optimal.

Let us now consider the first part of the lemma. Assume to the contradiction that there exists an optimal schedule in which the processing times of the fixed operations on the second machine are not equal to each other. In such a solution, there exists at least one occurrence such that $f_k^2 \ne f_{k+1}^2$ for adjacent parts k and $k+1$, $k \in [1,n-2]$.

We need to analyze the following cases:

1. $f_k^2 < f_{k+1}^2$: For this case we can construct a new schedule as $\hat{f}_k^2 = f_k^2+\delta$ and $\hat{f}_{k+1}^2 = f_{k+1}^2-\delta$ for $0 \le \delta \le (f_{k+1}^2-f_k^2)/2$ and all other processing times and assignments remain the same. As a consequence of this change, $\hat{C}_{k,2} \ge C_{k,2}$ and $\hat{C}_{k+1,2} = C_{k+1,2}$. Therefore, this new schedule is feasible and from Property 1, the cost does not increase. A similar procedure can be repeated for all adjacent pairs of non-identical processing times until all of them become equal.
2. $f_k^2 > f_{k+1}^2$: We will consider the following subcases:
   2.1. $x_{k+2} = 0$: As a consequence of Lemma 3 we also know that $x_k = x_{k+1} = 0$. From Case 1 and from the reversibility property mentioned above, we know that $f_{k+1}^1 \le f_{k+2}^1$. Additionally, from Lemma 2 we know that $C_{k+2,1} \le C_{k+1,2}$. Therefore, $s_k < s_{k+1}$ must be satisfied. Now, we can change the processing time values as $\hat{s}_k = s_k+\delta$, $\hat{s}_{k+1} = s_{k+1}-\delta$, $\hat{f}_k^2 = f_k^2-\delta$, and $\hat{f}_{k+1}^2 = f_{k+1}^2+\delta$, for $0 < \delta \le \min\{(f_k^2-f_{k+1}^2)/2, (s_{k+1}-s_k)/2\}$ to get a new schedule. This change does not affect the completion times of the parts and thus the new schedule is still feasible. From Property 1, its cost is not greater than the previous one.
   2.2. $x_{k+1} = 0$, $x_{k+2} = 1$: As a consequence of Lemma 3 we also know that $x_k = 0$. Under this case, if $s_{k+1} > s_k$, then we can construct a new schedule as $\hat{s}_k = s_k+\delta$, $\hat{s}_{k+1} = s_{k+1}-\delta$, $\hat{f}_k^2 = f_k^2-\delta$, and $\hat{f}_{k+1}^2 = f_{k+1}^2+\delta$, for $0 < \delta \le \min\{(f_k^2-f_{k+1}^2)/2, (s_{k+1}-s_k)/2\}$. This new schedule has the same completion times of parts with a smaller cost (Property 1). On the other hand, if $s_{k+1} \le s_k$, then $f_k^2+s_k > f_{k+1}^2+s_{k+1}$. From Lemma 2 we have $C_{k,2} \ge C_{k+1,1}$. Let us first consider $C_{k,2} = C_{k+1,1}$ case. From Lemma 2, $C_{k,1} \le C_{k-1,2}$. As a consequence, $f_{k+1}^1 \ge f_k^2+s_k > f_{k+1}^2+s_{k+1}$. Furthermore, since from Lemma 2 we have $C_{k+1,2} \ge C_{k+2,1}$, $f_{k+1}^2+s_{k+1} \ge f_{k+2}^1+s_{k+2}$. Therefore, we have $f_{k+2}^1 < f_{k+1}^1$. We can construct a new schedule as $\hat{f}_{k+1}^1 = f_{k+1}^1-\delta$ and $\hat{f}_{k+2}^1 = f_{k+2}^1+\delta$ for $0 \le \delta \le (f_{k+1}^1-f_{k+2}^1)/2$ where all other processing times and assignments remain the same. This new schedule is still feasible and from Property 1 the cost of this new schedule is not greater. Let us now consider the case $C_{k,2} > C_{k+1,1}$. We can construct a new schedule as $\hat{f}_k^2 = f_k^2-\delta$ and $\hat{f}_{k+1}^2 = f_{k+1}^2+\delta$, for $0 < \delta \le \min\{(f_k^2-f_{k+1}^2)/2, C_{k,2}-C_{k+1,1}\}$. In this new schedule $\hat{C}_{k,2} < C_{k,2}$ but $\hat{C}_{k+1,2} = C_{k+1,2}$. Therefore, it is feasible and from Property 1, it has a smaller cost value.
   2.3. $x_k = 0$, $x_{k+1} = 1$: As a consequence of Lemma 3 we also know that $x_{k+2} = 1$. If $s_{k+1} > s_{k+2}$, then we can construct a new schedule as $\hat{s}_{k+2} = s_{k+2}+\delta$, $\hat{s}_{k+1} = s_{k+1}-\delta$, $\hat{f}_k^2 = f_k^2-\delta$, and $\hat{f}_{k+1}^2 = f_{k+1}^2+\delta$, for $0 < \delta \le \min\{(f_k^2-f_{k+1}^2)/2, (s_{k+1}-s_{k+2})/2\}$. This new schedule has the same completion times of parts with a smaller cost (Property 1). On the other hand, if $s_{k+1} \le s_{k+2}$, then $f_{k+1}^1+s_{k+1} \le f_{k+2}^1+s_{k+2}$. This is because, from case 1 of the proof and from the reversibility property, we have $f_{k+1}^1 \le f_{k+2}^1$. From Lemma 2, we have $C_{k,2} \ge C_{k+1,1}$. Let us first consider the case $C_{k,2} = C_{k+1,1}$. Since $C_{k,1} \le C_{k-1,2}$ from Lemma 2, in order to have this, we must have $f_{k+1}^1+s_{k+1} \ge f_k^2+s_k$. Furthermore, in order to have $C_{k+1,2} \ge C_{k+2,1}$ as stated in Lemma 2, we must have $f_{k+1}^2 \ge f_{k+2}^1+s_{k+2}$. Combining these with $f_k^2 > f_{k+1}^2$ and $s_{k+1} \le s_{k+2}$, we have $f_{k+2}^1 < f_{k+1}^1$. This result is identical to the previous case where we showed how to construct a new schedule with a smaller cost. The remaining case where $C_{k,2} > C_{k+1,1}$ is also handled in the previous case and is not repeated here.
   2.4. $x_k = 1$: As a consequence of Lemma 3 we also know that $x_{k+1} = 1$ and $x_{k+2} = 1$. The arguments in this case is identical to the previous one hence left to the reader. $\square$

This lemma proves that there exists an optimal schedule in which the processing times of the fixed operations of all parts except the last part are identical to each other on the second machine. The first one of the following two corollaries of the above lemma is a direct consequence of the reversibility property and the other one can easily be proved similarly.

Corollary 1. *In the optimal schedule $f_j^1 = f_k^1$ for $2 \le j,k \le n$ and $f_1^1 \le f_j^1$, for $2 \le j \le n$.*

Let M^i denote the set of operations assigned to machine i. More formally, $M^i=\{j: x_j=2-i\}$, $i=1,2$.

Corollary 2. *In the optimal schedule $s_j=s_k$ for $j,k\in M^i$ for $i=1, 2$.*

Taking advantage of this corollary, s^1 and s^2 can be used to denote the processing time of flexible operations on machines 1 and 2, respectively. Let $N_1=\{2,3,\ldots,n\}$ and $N_2=\{1,2,\ldots,n-1\}$ and let p^i denote the processing time value on machine i which is equal for parts $j\in N_i$, $i=1, 2$. Also let $q^1=f_1^1$ and $q^2=f_n^2$. As a summary of Lemma 4 and Corollaries 1 and 2, we can write the following:

1. $q^i\le f_j^i=p^i$, $\forall j\in N_i$, $i=1, 2$
2. $s_h=s_j$, $\forall h,j\in M^i$, $i=1, 2$.

3.1. Complexity of the problem

In this study, our aim is to determine the assignment of the flexible operations to the machines and processing times for each operation to minimize the total nonlinear manufacturing cost and makespan simultaneously. Although we have proposed several theoretical properties of the problem that could be quite useful to reduce the search space significantly, we will show below that the computational complexity of our joint problem still remains an open problem.

Let us relax Constraints (14)–(19) (including (8)) in Model 2 along with any set of constraints required for linearization purposes. These are the non-interference and precedence constraints satisfying that a part can start on machine $i=1, 2$ if the processing of the previous part is completed and the processing of a part can start on machine 2 if it is completed on the first machine, respectively. Furthermore, let us assume that there is no unforced idle time in the optimal schedule. For this reduced problem, using Lemmas (1), (3), and (4), the optimal solution is attained if the following two equations are satisfied (from Lemma 4 let $f_j^1=f^1$, for $j=2,3,\ldots,n$ and $f_j^2=f^2$ for $j=1,2,\ldots,n-1$):

$$f_1^1+(n-1)f^2+rs^1+f_n^2=\epsilon \tag{23}$$

$$f_1^1+(n-1)f^1+(n-r)s^2+f_n^2=\epsilon \tag{24}$$

Moreover let us fix r due to Lemma 3, and solve the following relaxed model for a set of different r values, i.e. $r=1,\ldots,n$, to minimize the total manufacturing cost for a given makespan value (ϵ):

$$\textbf{Model 3}:\ \text{Min}\quad Z_1=\left(\sum_{i=1}^{2}(f_i^i+(n-1)f^i)+(rs^1+(n-r)s^2)\right)\cdot O + \sum_{i=1}^{2}((f_i^i)^b+(n-1)(f^i)^b)\cdot A^i+(r(s^1)^b+(n-r)(s^2)^b)\cdot A^s \tag{25}$$

$$\text{s.t.}\quad f_1^1+(n-1)f^2+rs^1+f_n^2=\epsilon \tag{26}$$

$$f_1^1+(n-1)f^1+(n-r)s^2+f_n^2=\epsilon \tag{27}$$

$$f_i^i,f^i\ge f_l^i\quad\forall i \tag{28}$$

$$s^i\ge s_l\quad\forall i \tag{29}$$

This relaxed model has a nonlinear objective function with six variables. Additionally, it has two equality constraints and bounding constraints for each decision variable. In Lemma 4, we proved that the processing times of all parts except the first (last) one are identical to each other on the first (second) machine in the optimal solution ($f_j^1=f^1$, for $j=2,3,\ldots,n$ and $f_j^2=f^2$ for $j=1,2,\ldots,n-1$). Examples 1 and 2 both show that in an optimal solution we may have $f_1^1<f_j^1,\forall j\ne 1$. The following is another example which also shows that in an optimal solution we may have $f_n^2<f_j^2,\forall j\ne n$.

Example 3. Let us assume we are given the following parameters: $n=7$, $O=0.58$, $A_1=12.7$, $A_2=14.4$, $A_3=5.9$, $b=-1.5$, $f_l^1=1.4$, $f_l^2=1.6$, and $s_l=1.8$. When the model is solved for $\epsilon=25.1$, in the optimal solution to this problem, the flexible operations are assigned to the second machine for the first three parts and to the second machine for the remaining parts. The processing times are $f_1^1=1.927$, $f_j^1=2.324$, $f_j^2=2.611$, $f_n^2=2.027$, $s^1=1.8$, and $s^2=1.827$.

In order to solve the relaxed model above, which is a continuous nonlinear resource allocation problem, there are mainly two approaches [10]: (i) First class of algorithms are based on KKT conditions. Since the objective is the minimization of a convex function, the KKT conditions can be used. (ii) Second class of algorithms are called pegging algorithms in which the bound conditions given in Eqs. (28) and (29) are relaxed and using the Lagrange multipliers some variable values are fixed at each iteration.

For a single constraint case, if the objective function in a nonlinear resource allocation problem is quadratic and the constraint is linear, then the exact solution can be found. Otherwise, as it is the case with the above formulation, it is not possible to find a closed form solution. The solution procedure will be an infinitely convergent procedure even when the bound constraints are relaxed [10]. One should implement an iterative algorithm like Bisection or Golden Section search to determine approximate values of the multipliers that solves a system of nonlinear equations. Such a solution algorithm can only provide approximate values.

Since we conjecture that a polynomial time solution may not exist even for the relaxed problem presented in Model 3, it is justifiable to develop a heuristic algorithm for the joint problem, e.g., Model 2, as discussed in the next section.

3.2. Processing time determination subproblem

Until this point, we know the relation between the processing times of the fixed operations on any one of the machines for different parts. The following lemma determines the relation between the processing times of the fixed operations on a machine and the processing times the flexible operations assigned to the same machine. For the definition we need the following sets:

$$J_1^i=\{j: f_u^i\ge f_j^{i^*}>f_l^i\}\quad\text{and}\quad J_2^i=\{j: j\ne 1, f_j^{i^*}=f_l^i\},\ i=1,2$$
$$J_1^s=\{k: s_u\ge s_k^*>s_l\}\quad\text{and}\quad J_2^s=\{k: s_k^*=s_l\}$$

Here, J_1^i and J_1^s are the sets of parts whose processing times that have a greater value than their lower bounds and J_2^i and J_2^s are the sets of parts whose processing times are at their lower bounds. M^i is the set of parts for which the flexible operations are assigned to machine i.

Lemma 5. *In an optimal solution to the problem, let f^{1^*}, f^{2^*}, and s^* denote the optimal processing time vectors. Then the following conditions hold*:

1. $\partial F^i(f_j^{i^*})\le\partial S(s_k^*),\forall j\in J_1^i,\ \forall k\in J_2^s\cap M^i$, $i=1,2$.
2. $\partial S(s_k^*)\le\partial F^i(f_j^{i^*}),\ \forall k\in J_1^s\cap M^i,\ \forall j\in J_2^i$, $i=1,2$.
3. $\partial F^i(f_j^{i^*})=\partial S(s_k^*),\ \forall j\in J_1^i,\ \forall k\in J_1^s\cap M^i$, $i=1,2$.

Proof. From Lemma 4 and Corollary 1, we know that the fixed operations on a machine have the same value except the first part on the first machine and the last part on the second machine. Also, from Corollary 2, the processing times of the flexible

operations assigned to the same machine have the same values. In order to prove this lemma, let us assume to the contradiction of the third case that there is an optimal schedule in which $\exists j \in J_1^i$ and $\exists k \in J_1^s \cap M^i$, $i=1$ or 2 such that $\partial F^i(f_j^{i^*}) \neq \partial S(s_k^*)$. Let us consider without loss of generality that $\partial F^i(f_j^{i^*}) > \partial S(s_k^*)$. This means that the contribution (in terms of reduction) of an increase in the value of s_k^* to the total cost is greater than that of $f_j^{i^*}$. Additionally, $\exists \delta$ such that $\partial F^i(f_j^{i^*}-\delta) \geq \partial S(s_k^*+\delta)$. Therefore, we can construct a new schedule as $\hat{f}_j^i = f_j^{i^*}-\delta$ and $\hat{s}_k = s_k^*+\delta$ which is still feasible since the completion time of this part is not changed after this modification. Since the derivative of the cost function S at point $s_k^*+\delta$ is still less than the derivative of F^i at point $f_j^{i^*}-\delta$, the resulting schedule has a smaller cost. This contradicts with the old solution being optimal. Therefore, the third case is proved.

If $\forall j \in J_2^i$ in the optimal solution, which means that $f_j^{i^*} = f_l^i$, then $f_j^{i^*}-\delta$ is not possible. Therefore, for this case, $\partial S(s_k^*) \leq \partial F^i(f_j^{i^*})$. This completes the proof of Case 2. The proof of Case 1 is identical to this one and left to the reader. □

Derivatives show the contribution of a change in the processing time to the manufacturing cost, so the processing time values of variables are determined by comparison of derivatives of the cost functions with respect to the processing times.

The proposed algorithm starts with setting the processing time variables to their lower bounds and determines the assignment of flexible operations. The corresponding solution is represented as point A in Fig. 1. If there is any idle time on any one of the machines, applying the following rule to that machine reduces the cost without increasing the makespan. This idea was highlighted in Example 1 in Section 2.2. This new solution is denoted as point B in Fig. 1 and this new point dominates point A. Let $f_j^i new$ and $s^{i^{new}}$ denote the new values of operation i of job j and flexible operations of jobs processed on machine i, respectively. The algorithm determines these using the current values f_j^i and s^i.

Rule 1: Let I_i represents the idle time on machine i after the assignments are made when the processing times are at their lower bounds. r_i denote the number of flexible operations assigned to machine i, where $r_1+r_2=n$. Then as a direct consequence of Lemma 5, this idle time can be covered either by increasing only f_j^i or s^i or both. In the last case, after the new processing time values are determined, the derivatives of the cost functions will be equal to each other. Therefore, one of the following conditions holds:

1. $s^{i^{new}} = s_l$ and $f_j^i new = p^i + I_i/(n-1)$ for $j \in N_i$, $i=1, 2$.
2. $s^{i^{new}} = s^i + I_i/r_i$ and $f_j^i new = f_l^i$ for $j \in N_i$, $i=1, 2$.
3. $s^{i^{new}} = \dfrac{I_i + (n-1)\cdot p^i + r_i \cdot s^i}{r_i + (n-1)\cdot \sqrt[b-1]{\dfrac{A_s}{A_i}}}$

   and

   $f_j^i new = \dfrac{I_i + (n-1)\cdot p^i + r_i \cdot s^i}{n-1+r_i \cdot \sqrt[b-1]{\dfrac{A_i}{A_s}}}$ for $j \in N_i, i=1,2$

The first (second) case distributes the total idle time among the fixed (flexible) processing times. The last case is found by solving the following system of equations where the derivatives of the cost functions are made equal to each other.

$$I_i = (n-1)\cdot(f_j^i new - p^i) + r_i \cdot (s^{i^{new}} - s^i) \quad \text{for } j \in N_i,\ i=1,2$$

$$A^i \cdot b \cdot (f_j^i new)^{b-1} = A^s \cdot b \cdot (s^{i^{new}})^{b-1} \quad \text{for } j \in N_i,\ i=1,2$$

We generate another nondominated solution by setting all of the processing time values to their upper bounds and determine the assignment of flexible operations afterwards. This solution is represented as point C in Fig. 1. After determining these two nondominated points, we generate a set of discrete nondominated points in between them on the efficient frontier starting from the nondominated solution B. This is done by increasing the upper limit ϵ in Constraint (12) by a predetermined increment, δ. This increment may change the assignment of the flexible operations as well as the processing times. Therefore, we have to solve two challenging optimization problems simultaneously. In the following Rule 2, we demonstrate that the new optimum processing time values could be found in polynomial time using the proposed closed form expressions for a given δ if the assignments remain the same. In this rule, the increment is first applied to one of the machines, then the corresponding schedule is found for the other machine.

Rule 2: Let $f_j^1 new$ and $s^{1^{new}}$ ($f_j^2 new$ and $s^{2^{new}}$) be the values of the 1st (2nd) operation of job j and flexible operations of jobs processed on machine 1 (machine 2) at the next point on the efficient frontier, respectively. Then, one of the following conditions hold as long as the assignment of flexible operations remains the same.

1. If $\min_i\{\partial F^i(p^i),\ \partial S(s^i)\} = \partial F^k(p^k)$, then the new processing times on machine k are determined by either Case (a) or (c) by setting $i=k$, and the new processing times on the other machine is found by one of the cases (d), (e), or (f) by setting $i=(3-k)$.
2. If $\min_i\{\partial F^i(p^i),\ \partial S(s^i)\} = \partial S(s^k)$, then the new processing times on machine k are determined by either Case (b) or (c) by setting $i=k$, and the new processing times on the other machine is found by one of the cases (d), (e), or (f) by setting $i=(3-k)$.

   (a) $f_j^i new = (\delta + \sum_{h=1}^n f_h^i)/n$, $\forall j$ and $s^{i^{new}} = s^i$.

   (b) $f_j^i new = f_j^i$, $\forall j$ and $s^{i^{new}} = (\delta + r_i \cdot s^i)/r_i$.

   (c)
   $$f_j^i new = \frac{\delta + \sum_{h=1}^n f_h^i + r_i \cdot s^i}{n + r_i \cdot \sqrt[b-1]{\dfrac{A^i}{A^s}}} \quad \forall j$$
   and
   $$s^{i^{new}} = \frac{\delta + \sum_{h=1}^n f_h^i + r_i \cdot s^i}{r_i + n \cdot \sqrt[b-1]{\dfrac{A^s}{A^i}}}$$

   (d) $f_j^i new = (\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + \sum_{h \in N_i} f_h^i)/(n-1)$, $j \in N_i$, $q^{i^{new}} = q^i$ and $s^{i^{new}} = s^i$.

   (e) $f_j^i new = f_j^i$, $\forall j$ and $s^{i^{new}} = (\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + r_i \cdot s^i)/r_i$.

   (f)
   $$f_j^i new = \frac{\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + \sum_{h \in N_i} f_h^i + r_i \cdot s^i}{n-1 + r_i \cdot \sqrt[b-1]{\dfrac{A_{(3-i)}}{A_s}}} \quad \forall j$$
   $$q^{i^{new}} = q^i \quad \text{and} \quad s^{i^{new}} = \frac{\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + \sum_{h \in N_i} f_h^i + r_i \cdot s^i}{r_i + (n-1)\cdot \sqrt[b-1]{\dfrac{A_s}{A_{(3-i)}}}}$$

In this rule, Cases (a)–(c) distribute δ to the processing times on the selected machine, whereas conditions (d)–(f) calculate the new processing time values on the other machine as the second step. From these, Case (a) uses only the fixed processing times to satisfy the increase, and Case (b) uses only the flexible operations on the selected machine for this purpose (Lemma 4 and

Corollaries 1 and 2). In Case (c), both fixed and flexible operations are used such that at the end the derivatives of the cost functions are equal to each other (Lemma 5). After the processing times on the selected machine are determined, Cases (d)–(f) are used to determine the new processing times on the other machine. Similar to the previous ones, Case (d) uses only the fixed operations, Case (e) uses only the flexible operations, and Case (f) uses both of these operations such that the derivatives of the cost functions become equal to each other. SubroutineF^i shows the use of the first part of this rule as an algorithm. The second one can also be written similarly.

SubroutineF^i. Determination of new processing times for Case 1 of Rule 2.

1: **if** $\min\{\partial F^i(f_j^i),\partial S(s^i)\} = \partial F^i(f_j^i)$ **then**
2: **if** $A \cdot b \cdot ((\delta + \sum_{j=1}^{n} f_j^i)/n)^{b-1} \le \partial S(s^i)$ **then**
3: Calculate $f_j^i new$ and $s^{i^{new}}$ using (a).
4: **else**
5: Calculate $f_j^i new$ and $s^{i^{new}}$ using (b).
6: **end if**
7: **if** $\partial F^{(3-i)}(f_j^{(3-i)}) \le \partial S(s^{(3-i)})$ **then**
8: **if** $A \cdot b \cdot ((\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + \sum_{j \in N_{(3-i)}} f_j^{(3-i)})/(n-1))^{b-1} \le \partial S(s^{(3-i)})$ **then**
9: Calculate $f_j^{(3-i)} new$ and $s^{(3-i)^{new}}$ using (c).
10: **else**
11: Calculate $f_j^{(3-i)} new$ and $s^{(3-i)^{new}}$ using (e).
12: **end if**
13: **else**
14: **if** $A \cdot b \cdot ((\delta - (q^{(3-i)^{new}} - q^{(3-i)}) + r_{(3-i)} \cdot s^{(3-i)})/r_{(3-i)})^{b-1} \le \partial F^{(3-i)}(f_j^{(3-i)})$ **then**
15: Calculate $f_j^{(3-i)} new$ and $s^{(3-i)^{new}}$ using (d).
16: **else**
17: Calculate $f_j^{(3-i)} new$ and $s^{(3-i)^{new}}$ using (e).
18: **end if**
19: **end if**
20: **end if**

4. Proposed algorithm

In this section, we will present the detailed steps of the proposed algorithm, named as EFFLOW Algorithm. Our goal is to generate a set of solutions (such a way whose image approximates the efficient set in the objective space) in a short time to provide a good approximation of the efficient set that will allow the decision maker to choose a good compromise solution. As will be proved in Lemma 6, any two solutions generated by the algorithm cannot dominate each other. However, there may be another solution generated by an exact algorithm which can dominate the solutions generated by the proposed algorithm. Later on, the solution quality of the proposed approach will be compared with available exact efficient solutions.

EFFLOW algorithm starts with an initial schedule at which the processing times are at their lower bounds and the optimal assignment of flexible operations to machines are determined according to the procedure developed by Crama and Gultekin [1]. As we mentioned before, if there exists idle times in the solution obtained via this procedure, the idle times are eliminated using Rule 1 to get the initial nondominated solution.

Starting from this initial solution, in order to generate the next nondominated solution the makespan upper bound (ϵ) is increased by δ, the corresponding allocation of the flexible operations and the processing time values are decided according to this new makespan value. This procedure is repeated until ϵ becomes identical to the upper bound for the makespan found by setting all processing times to their upper bounds. Let r_l (r_u) and C_{maxl} (C_{maxu}) represent the number of flexible operations assigned to machine 1 and the value of makespan, respectively, when the processing time variables are equal to their lower (upper) bounds.

Main Algorithm. EFFLOW Algorithm.

Input: f_l^1, f_l^2, s_l, f_u^1, f_u^2, s_u, F^1, F^2, S, and δ
Output: A number of nondominated (C_{max}, Total Cost) pairs with corresponding, r, f_j^1, f_j^2, s^1, s^2, $T_{j,m}$ values for each point
1: Set processing times to their lower bounds
2: Compute r_l and C_{maxl}
3: Assign the flexible operations to the machines according to r_l
4: Compute starting times of jobs on each machine, $T_{j,m}$
5: **if** $T_{n,1} + f_n^1 + x_n \cdot s_1 \le T_{n,2}$ **then**
6: $I_1 \leftarrow T_{n,2} - (T_{n,1} + f_n^1 + x_n \cdot s^1)$
7: Cover idle time on machine 1 using Rule 1
8: **end if**
9: **if** $T_{n-1,2} + f_1^2 + (1-x_{n-1}) \cdot s^2 \le T_{n,1} + f_n^1 + x_n \cdot s^1$ **then**
10: $I_2 \leftarrow T_{n,1} + f_n^1 + x_n \cdot s^1 - (T_{n-1,2} + f_1^2 + (1-x_{n-1}) \cdot s^2)$
11: Cover idle time on machine 2 using Rule 1
12: **end if**
13: Compute r, r_u, C_{maxu}, and $t = \lfloor (C_{\max u} - C_{\max l})/\delta \rfloor$
14: **for** $j=1$ to t **do**
15: **if** $\min\{\partial F_j^1, \partial F_j^2, \partial S^1, \partial S^2\} = \partial F_j^1$ or ∂S^1 **then**
16: Use Subroutine Process-1 to determine allocation of flexible operations and processing time values
17: **else if** $\min\{\partial F_j^1, \partial F_j^2, \partial S^1, \partial S^2\} = \partial F_j^2$, or ∂S^2 **then**
18: Use Subroutine Process-2 to determine allocation of flexible operations and processing time values
19: **end if**
20: **end for**

EFFLOW algorithm gets the lower bounds of operation processing times as an input and initially assigns lower bounds to the processing time variables. Afterwards, the algorithm computes r_l and C_{maxl}. The allocation of flexible operations is assigned to a binary variable x_j according to r_l. x_j is assigned "1" if the flexible operation is assigned to the first machine, and "0" otherwise. According to the flexible operations allocation, starting times of jobs on each machine, $T_{j,m}$, are computed. At this instant, we get a schedule in which the processing times of operations of all jobs are at their lower bounds. At lines 5–8 and 9–12, the algorithm checks out for idle times on machines 1 and 2, respectively. If there exists any, it covers idle times as discussed in Rule 1. After that, the algorithm computes r_u and C_{maxu}. In the loop of EFFLOW algorithm, makespan value is increased by δ to determine points on the efficient frontier. Therefore, for a specified value of δ, the algorithm generates a total of $t = \lfloor (C_{maxu} - C_{maxl})/\delta \rfloor$ solutions in between C_{maxl} and C_{maxu} values on the efficient frontier. Either δ or t can be given as an input to the algorithm. Allocation of flexible operations and processing time values are determined using Subroutines Process-1 or Process-2 till we reach to C_{maxu}. Since the subroutines are similar, we only present Subroutine Process-1 due to space limitations.

Subroutine Process-1. Increasing the total processing time on machine 1 by δ.

Increase the total processing time on machine 1 by δ using Rule 2
$I_2 \leftarrow \delta - \Delta(f_1^1)$
Cover idle time on machine 2 using Rule 1
Check upper bounds of variables for any violation and make necessary corrections
Compute manufacturing cost
if $r > r_u$ **then**
  $I_2 \leftarrow \delta - \Delta(f_1^1)$
  if $s_l - I_2 \le (n-r)\cdot(s^2 - s_l) + (n-1)\cdot(f_1^2 - f_l^2)$ **then**
    $r \leftarrow r-1$
    $I_2 \leftarrow I_2 - s^2$
    Cover idle time on machine 2 using Rule 1
    $I_1 \leftarrow s^1$
    Cover idle time on machine 1 using Rule 1
    Check upper and lower bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
if $r < r_u$ **then**
  if $s_l - \delta \le r\cdot(s^1 - s_l) + (n-1)\cdot(f_n^1 - f_l^1)$ **then**
    $r \leftarrow r+1$
    $I_1 \leftarrow \delta - s^1$
    Cover idle time on machine 1 using Rule 1
    $I_1 \leftarrow \delta + s^2$
    Cover idle time on machine 2 using Rule 1
    Check upper and lower bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
Output: the minimum manufacturing cost, assignment of flexible operations, and associated processing times

The Subroutines Process-1 and Process-2 start with increasing makespan by "δ" on machines 1 and 2, respectively. Increasing makespan on the selected machine incurs an idle time on the other machine. $\Delta(f_1^1)$ and $\Delta(f_n^2)$ represents $(f_1^1 new - f_1^1)$ and $(f_n^2 new - f_n^2)$, respectively. $f_1^1 new$ is the processing time value of f_1^1 after increasing the total processing time on machine 1 by δ and $f_n^2 new$ is the processing time value of f_n^2 after increasing the total processing time on machine 2 by δ. Therefore, the subroutines cover the idle time using the idea presented in Rule 1. Afterwards, any upper bound violations of processing time variables are investigated. If there exists an operation processing time which is greater than its upper bound, it is assigned to its upper bound, and the total reduction in the value of this variable is distributed to the other operations on the same machine as an increase. If any one of these processing times also becomes greater than its upper bound after increasing its processing time, it is also assigned to its upper bound. In such a case all operations on this machine hit their upper bounds, so we cannot make any increase of processing times on this machine any more and some idle time remain uncovered. After this correction, manufacturing cost is computed with the new processing time values using the following formula:

$$\text{Manufacturing Cost} = (f_1^1 + (n-1)\cdot f_n^1 + (n-1)\cdot f_1^2 + f_n^2 + r\cdot s^1 + (n-r)\cdot s^2)\cdot O + ((f_1^1)^b + (n-1)\cdot(f_1^1)^b)\cdot A^1 + ((n-1)\cdot(f_1^2)^b + (f_n^2)^b)\cdot A^2 + (r\cdot(s^1)^b + (n-r)\cdot(s^2)^b)\cdot A^s$$

Subroutines Process-1 and Process-2 compare r value with r_u to check whether changing the assignment of one of the flexible operations reduces the cost or not. If the number of flexible operations on machine 1, represented as r, is equal to the number of flexible operations on machine 1 when all processing times are at their upper bounds, represented as r_u, the assignment of flexible operations should not change and remain the same till makespan reaches to C_{maxu}. In this case, the subroutines terminate by returning the value of manufacturing cost and associated processing time values of operations of jobs. If r is greater than r_u, the last flexible operation on machine 1 is assigned to the machine 2, if possible. This assignment may not always be possible because of makespan limitation constraint, so the subroutines investigate whether a flexible operation can fit on the machine 2 while all processing time variables are at their lower bounds. If it is proved to be feasible, the assignment of flexible operation and the values of processing times change accordingly. This may cause some of the processing time variables to decrease or increase. If any processing time value exceeds its upper bound, the necessary modification is made as already mentioned. On the other hand, in case of a lower bound violation of an operation processing time, the variable is assigned to its lower bound, and the total increase in the value of this variable is distributed to the other operations on the same machine as a reduction. If this reduction violates the lower bound of the processing time for any other operation, its processing time is set to the lower bound. In the worst case, all processing times on the machine hit to their lower bounds. Afterwards, manufacturing cost is computed. For the case of r being smaller than r_u, the first flexible operation on machine 2 is assigned to the machine 1, if possible. The same procedure is applied with the former case. Finally the subroutines output the minimum manufacturing cost, the assignment of the flexible operations, and the associated processing time values.

EFFLOW algorithm generates a set of nondominated solutions that are equally spaced on the efficient frontier. The following lemma presents this important property of the algorithm.

Lemma 6. *Any two solutions generated by the EFFLOW algorithm cannot dominate each other.*

Proof. Let $Z_1^\star = \sum_{j=1}^{n}\sum_{i=1}^{2}(F^i(f_j^{i^\star}) + S(s_j^\star))$ be the objective function value with processing time vectors $f^\star$ and $s^\star$ and let $C_{max}^\star = T_{n,2} + f_n^{2^\star} + s_n^\star\cdot(1-x_n)$ be the makespan value generated by the algorithm. A new solution is generated by incrementing the makespan value. Let $\hat{Z}_1 = \sum_{j=1}^{n}\sum_{i=1}^{2}(F^i(\hat{f}_j^{i}) + S(\hat{s}_j))$ be the objective function value with processing time vectors $\hat{f}$ and $\hat{s}$ and let $\hat{C}_{max} = T_{n,2} + \hat{f}_n^2 + \hat{s}_n\cdot(1-x_n)$ be the makespan value of this new solution. Since $\hat{C}_{max} > C_{max}^\star$, at least one of the processing time values has increased in the new solution. Since the cost function is decreasing with respect to processing times, we have $\hat{Z}_1 < Z_1^\star$. This proves that the solutions generated by the EFFLOW algorithm cannot dominate each other and a new nondominated solution is generated at each iteration of the algorithm. □

Moreover, time complexity of the EFFLOW algorithm is given below. Let t denote the total number of efficient solutions that is desired to be generated by the algorithm. Using this, the increment value δ can be calculated as $\delta = (C_{maxu} - C_{maxl})/t$.

Proposition 1. *Time complexity of the EFFLOW algorithm is* $\mathcal{O}(tn)$.

Proof. In EFFLOW algorithm, computation of r_l and r_u requires constant time. Once the assignments of the flexible operations are determined, computation of the starting times of all jobs and hence the computation of C_{maxl} and C_{maxu} has time complexity $\mathcal{O}(n)$. In Rule 1 and Rule 2, new processing times of all parts are determined and thus their time complexity are both $\mathcal{O}(n)$. Since

the Subroutines Process-1 and Process-2 use Rule 1 and Rule 2 without any loops, their time complexity are also $\mathcal{O}(n)$. The main loop of the EFFLOW algorithm is repeated t times to generate t solutions and within each repetition either Subroutine Process-1 or Process-2 is performed once. Therefore, time complexity of the EFFLOW algorithm appears to be $\mathcal{O}(tn)$. □

5. Computational results

In this section, we perform a computational study to test the performance of our proposed algorithm by comparing it with the mathematical formulations, Models 1 and 2. Mixed integer nonlinear programs are formulated in GAMS 22.0. One of the alternatives to solve MINLP formulations is to call BARON solver. This solver guarantees to provide global optima under fairly general assumptions using deterministic global optimization algorithms of the branch-and-bound type. However the CPU time requirement of BARON may be too limiting and it may not be possible to solve problems in a reasonable time, so we run BARON with a time limit of 1000 s. Another alternative for generating good quality solutions in smaller CPU times is to use DICOPT solver. DICOPT guarantees to converge only under certain convexity assumptions. Although the algorithm has provisions to handle non-convexities, it does not necessarily obtain the global optimum. Due to CPU restrictions, it is not possible to run the mathematical model till the end by DICOPT. Therefore, we also run DICOPT with the same time limit as BARON. We observed that at some replications DICOPT cannot generate integer solutions but presents a relaxed solution with this time limit. On the contrary, no such points exist for BARON.

The proposed EFFLOW algorithm is coded in the C++ language and compiled with Gnu compiler. The DICOPT is ran on a computer with 3 GB memory and dual core Intel Pentium processor with 2.1 GHz CPU. However, due to licensing limitations, the BARON software is ran on a computer with 1294 MB memory and Pentium III 1133 MHz CPU with a time limit of 1000 s.

There are four experimental factors that can affect the efficiency of the algorithm as listed in Table 1. The experimental design is a 2^4 full factorial design. The factors A^1, A^2 and A^s are effective on the upper bounds of the processing times and on the manufacturing cost as presented before. We choose the factors from two different levels, which allows us to have different upper bounds for the processing times of operations and different cost functions for the operations. The Levels 1 and 2 will be represented as L and H, meaning low and high, in the following tables, respectively. The diversity of upper bounds changes the assignment of flexible operations which is expected to affect the efficiency of the EFFLOW algorithm. The factor n determines the size of the problem. When the problem size is large, it takes more CPU time to solve the problem, and the resulting cost and makespan objective values are expected to be high. Most of the exact algorithms need huge CPU times in case of large problem sizes, so our aim is to provide an algorithm that could provide high solution quality in small CPU times. The other parameters are selected randomly from the intervals of O=U[0.4, 0.8], $f_l^1 = U[1.2,1.7]$, $f_l^2 = U[1.4,1.9]$, s_l=U[1.6, 2.1], b=U[−1.7, −1.3], where U[a, b] is an uniform distribution in interval [a, b].

We took five replications for each factor combination and 25 different efficient point generation iterations for each replication resulting in $2^4 \cdot 5 \cdot 25 = 2000$ individual runs for the EFFLOW algorithm. The performance measures used in evaluating the experimental results are the percentage deviations of the EFFLOW algorithm from the mathematical formulations, Models 1 and 2, and the run times in CPU seconds.

The percent deviation of each run is calculated as $100 \cdot (Z_1^A - Z_1^B)/Z_1^B$, where Z_1^A represents the manufacturing cost value found by the EFFLOW algorithm and Z_1^B represents the solution of mathematical formulation reported by either DICOPT or BARON solvers for a given makespan value.

Because of the underlying algorithm of DICOPT, linearization of the constraints is thought to be helpful, so we also take runs of Model 2. Since Models 1 and 2 cannot be decided to be better than the other and at each iteration there exists efficient points at which the models improve each other, we present runs of both models with DICOPT. BARON runs are only taken for Model 1, because overall percent deviations of the EFFLOW algorithm from Models 1 and 2, solved by DICOPT, show that Model 1 performs better than Model 2.

5.1. Sample replication analysis

The points generated by the EFFLOW algorithm for one sample replication with a factor combination A^1=U[12, 15] , A^2=U[5, 8], A^s=U[5, 8], n=20, are plotted in Fig. 4 to indicate the shape of the efficient frontier of our bicriteria problem. Percent deviations of the proposed algorithm from Model 1 solved by DICOPT and BARON are plotted in Fig. 5a and b, respectively, to visualize the deviations. The first points (maximum makespan, minimum cost) and the last points (minimum makespan, maximum cost) in Fig. 5a and b correspond to the points B and C in Fig. 1, respectively.

As can be seen in Fig. 5a, while some of the deviations are positive, the others are negative. A positive deviation means that DICOPT finds a better solution than our proposed algorithm. On the other hand, a negative deviation means the EFFLOW algorithm can find a better

Table 1
Experimental factors.

Factors	Definition	Level 1	Level 2
A^1	Tool cost multiplier of first operation	U[5,8]	U[12,15]
A^2	Tool cost multiplier of second operation	U[5,8]	U[12,15]
A^s	Tool cost multiplier of flexible operation	U[5,8]	U[12,15]
n	Number of jobs	20	30

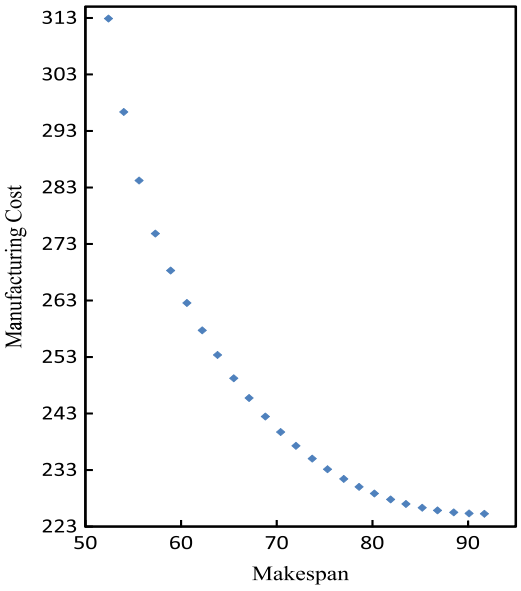

Fig. 4. Efficient frontier generated by the proposed algorithm for one replication.

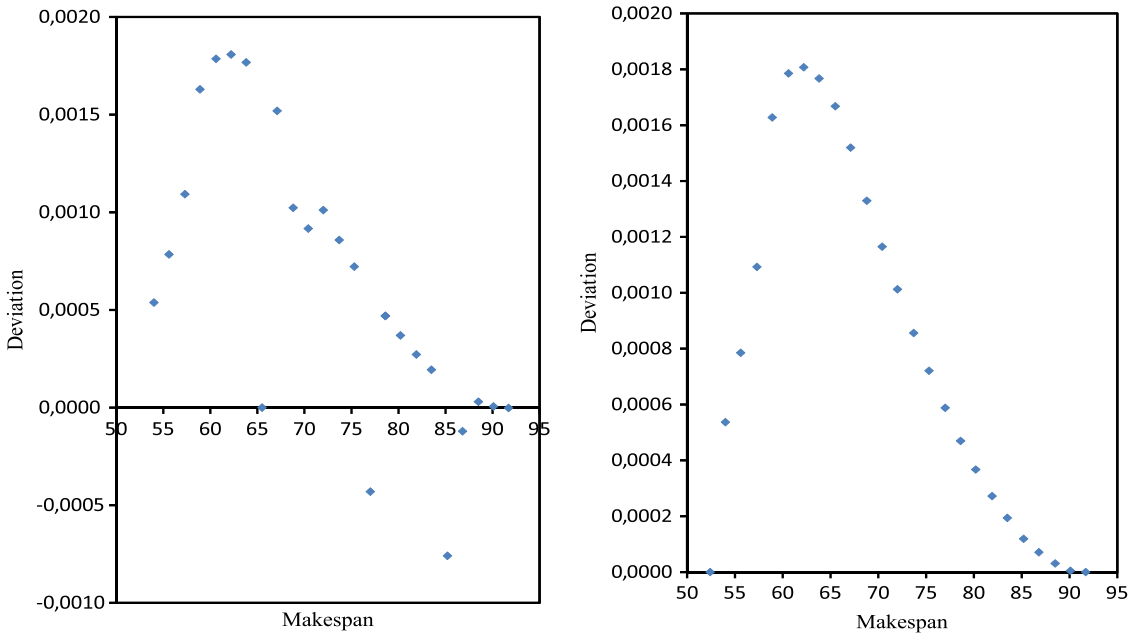

Fig. 5. Deviations on different regions of the efficient frontier. (a) DICOPT. (b) BARON.

Table 2
Percent deviations for each factor combination.

n	A^1	A^2	A^s	DICOPT						BARON		
				Model 1			Model 2			Model 1		
				Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
20	L	L	L	−2.72	0.05	0.30	−0.65	0.01	0.33	0	0.1	0.33
20	H	L	L	−1.95	0.05	0.40	−0.70	−0.01	0.39	−0.06	0.1	0.39
20	L	H	L	−0.94	0.07	0.37	−1.11	0.03	0.41	0	0.12	0.52
20	H	H	L	−1.62	0.11	0.42	−0.42	0.09	0.61	−0.05	0.09	0.33
20	L	L	H	−1.09	−0.03	0.33	−1.24	−0.02	0.33	−0.01	0.18	0.61
20	H	L	H	−2.12	0.05	0.55	−0.86	−0.03	0.55	−0.02	0.1	0.35
20	L	H	H	−3.44	−0.06	0.35	−0.92	−0.03	0.34	0	0.1	0.55
20	H	H	H	−3.75	0.06	0.41	−0.83	0.06	0.42	0	0.14	0.42
30	L	L	L	−5.22	−0.17	0.13	−0.84	−0.03	0.14	−0.03	0.04	0.17
30	H	L	L	−2.07	−0.03	0.38	−4.38	−0.20	0.25	−0.19	0.32	1
30	L	H	L	−1.40	0.04	0.22	−2.55	−0.06	0.26	−0.04	0.07	0.24
30	H	H	L	−1.71	−0.04	0.26	−0.57	0.02	0.32	−0.08	0.04	0.13
30	L	L	H	−5.95	−0.19	0.13	−0.71	−0.05	0.13	0	0.1	0.32
30	H	L	H	−0.47	0.06	0.28	−1.14	−0.14	0.29	−0.16	0.08	1.7
30	L	H	H	−3.54	−0.06	0.29	−1.42	−0.08	0.22	−0.08	0.07	0.29
30	H	H	H	−4.27	−0.04	0.30	−0.55	0.04	0.3	0	0.09	0.3

solution than DICOPT at that makespan value. This is possible since DICOPT does not necessarily obtain the global optimum. The minimum deviation, which is −0.0008, from Model 1 appears at a makespan value 85.2. The maximum deviation, which is 0.0018, from Model 1 appears at three different makespan values 60.6, 62.2, and 63.8. The deviations are small for smaller makespan values, then slightly increase through the mid-points of the efficient frontier. Deviations get smaller through the upper bound of the makespan.

When we consider the deviations of BARON in Fig. 5b, the minimum deviation, which is 0, appears at the first point and last three points generated. The maximum deviation, which is 0.0018, appears at three different makespan values 60.6, 62.2, and 63.8. The maximum deviation of the EFFLOW algorithm from Model 1 solved by BARON is the same as DICOPT and appears at the same points. As can be seen in the figure, all deviations are nonnegative. This means solver BARON performs better than both DICOPT and the EFFLOW algorithm. However, even the maximum deviation is very small, which indicates that the EFFLOW algorithm generates high quality solutions. Moreover, the deviations get smaller through the lower and upper bounds of the makespan.

5.2. Percent deviations

For the EFFLOW algorithm, the minimum, average, and maximum values of the percent deviations from mathematical formulations are given for all factor combinations in Table 2. Since we took five replications for each factor combination and each replication has 25 efficient points, average deviation considers these $25 \cdot 5 = 125$ points except the instances where the solvers are failed to find a feasible integer solution under the given time limit.

As mentioned before, the upper bounds of the processing times are affected by the factors A^1, A^2 and A^s. The assignment of flexible operations may also change with respect to the processing time values selected from different intervals which is

expected to affect the efficiency of the EFFLOW algorithm. According to the results of percent deviations for each factor combination presented, we may claim that the performance of the EFFLOW algorithm is independent of the level of the factors. The algorithm generates good quality solutions at both levels. Although the factor n mainly determines the size of the problem and CPU time to solve the problem, presented results indicate that the EFFLOW algorithm performs better for high level of number of jobs. When the number of jobs increases, the error tolerance in determination of processing time values and assignments of flexible operations may increase.

The percent deviations of the EFFLOW algorithm from Models 1 and 2 solved by DICOPT and from Model 1 solved by BARON are arranged according to each factor being at levels L and H, which are presented in Table 3. When mean deviations of the EFFLOW algorithm from DICOPT are considered, the algorithm performs better when A^1 and A^2 are at low levels and A^s is at high level with respect to both Models 1 and 2. The algorithm has the best performance when $n=30$ with respect to the minimums of min, mean and max deviations at this level. According to the deviations under the column of BARON, the EFFLOW algorithm again performs better for $n=30$ for BARON as for the DICOPT.

In Table 4, the overall percent deviations of all factor combinations along with the number of cases where DICOPT is failed to produce integer solutions (denoted by NNS) are presented. Mean deviations show that the EFFLOW algorithm improves both Models 1 and 2 solved by using DICOPT. DICOPT outputs better results for Model 1 than Model 2 when mean deviations are considered. The mean deviation of the EFFLOW algorithm from Model 1 solved by BARON is 0.0011, which indicates that the EFFLOW algorithm generates high quality solutions.

5.3. CPU times

CPU results of all factor combinations are presented for the EFFLOW algorithm, Models 1 and 2 solved by DICOPT in Table 5.

Table 3
Percent deviations for each factor level.

Factor	Level	DICOPT						BARON		
		Model 1			Model 2			Model 1		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
A^1	L	−5.95	−0.04	0.37	−2.55	−0.03	0.41	−0.08	0.10	0.61
	H	−4.27	0.03	0.55	−4.38	−0.02	0.61	−0.19	0.12	1.70
A^2	L	−5.95	−0.026	0.55	−4.38	−0.06	0.55	−0.19	0.13	1.70
	H	−4.27	0.01	0.42	−2.55	0.01	0.61	−0.08	0.09	0.55
A^s	L	−5.22	0.01	0.42	−4.38	−0.02	0.61	−0.19	0.11	1.00
	H	−5.95	−0.026	0.55	−1.42	−0.03	0.55	−0.16	0.11	1.70
n	L	−3.75	0.038	0.55	−1.24	0.01	0.61	−0.06	0.12	0.61
	H	−5.95	−0.054	0.38	−4.38	−0.06	0.32	−0.19	0.10	1.70

Table 4
Overall deviations.

Model 1 (DICOPT)				Model 2 (DICOPT)				Model 1 (BARON)		
Min	Mean	Max	NNS	Min	Mean	Max	NNS	Min	Mean	Max
−0.0595	−0.00003	0.0055	20	−0.0438	−0.00025	0.0061	33	−0.0019	0.0011	0.0170

Table 5
CPU times of mathematical models (DICOPT) and EFFLOW algorithm.

n	A^1	A^2	A^s	GAMS (DICOPT)						EFFLOW		
				Model 1			Model 2			Algorithm		
				Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
20	L	L	L	0.16	10.6	1000	0.19	6.11	115	0.0044	0.0072	0.0091
20	H	L	L	0.02	7.02	651	0.12	5.69	87	0.0029	0.0072	0.0107
20	L	H	L	0.01	26.7	1000	0.21	4.97	86	0.008	0.0112	0.0188
20	L	L	H	0.16	3.56	74	0.18	2.86	59	0.0044	0.0059	0.0085
20	H	H	L	0.02	9.94	805	0.01	5.69	113	0.0062	0.01	0.0143
20	L	H	H	0.08	6.96	594	0.16	6.09	235	0.005	0.0072	0.0125
20	H	L	H	0.06	9.18	748	0.14	2.6	33	0.005	0.008	0.01
20	H	H	H	0.16	2.85	79	0.2	6.04	150	0.0062	0.0123	0.0293
30	L	L	L	0.14	66.4	1000*	0.16	82.4	1000*	0.0044	0.0079	0.0158
30	H	L	L	0.14	90.14	1000*	0.2	129.71	1000*	0.0068	0.0115	0.0187
30	L	H	L	0.14	49.41	1000	0.17	88.72	1000*	0.0062	0.0105	0.0187
30	L	L	H	0.11	74.17	1000	0.2	24.92	1000*	0.0062	0.0076	0.0087
30	H	H	L	0.02	65.2	1000*	0.19	81.59	1000*	0.0081	0.0132	0.0187
30	L	H	H	0.09	57.37	1000*	0.16	40.76	1000*	0.0087	0.0157	0.03
30	H	L	H	0.2	81.24	1000*	0.23	72.62	1000*	0.0062	0.0115	0.0231
30	H	H	H	0.02	25.04	1000	0.22	40.54	1000*	0.0125	0.0181	0.0256

Table 6
Overall deviations for 1 s time limit.

n	Model 1 (DICOPT)				Model 2 (DICOPT)			
	Min	Mean	Max	NNS	Min	Mean	Max	NNS
20	−0.1062	−0.00016	0.0055	15	−0.0185	−0.00016	0.0061	31
30	−0.1867	−0.00162	0.0068	36	−0.0285	−0.00141	0.0030	35

Due to CPU restrictions, even for 30 jobs processed, it is not possible to run the mathematical model till the end by DICOPT. Therefore, we run the DICOPT with a time limit of 1000 s, which is the default. The same limit is also used for BARON. BARON solver did not stop before the time limit exceeded for all replications, which means every replication has a CPU time of 1000 s. Therefore, we do not present a CPU time requirement table for BARON.

The results indicate that the proposed algorithm can generate a high solution quality in a very small CPU time compared to DICOPT and BARON. As the number of jobs increases, the increase in the CPU time is very small for the proposed algorithm. As can be seen in Table 5, DICOPT cannot generate any integer solutions in 1000 s for some of the replications, represented as 1000*. Also for some of the replications, DICOPT stops due to 1000 s CPU restriction and reports the best integer solution. Solver BARON can generate integer solutions for all replications unlike DICOPT. According to the analysis in Section 5.2, BARON generates better solutions than DICOPT. However, the CPU requirement of BARON is much higher as compared with DICOPT. On the contrary, the EFFLOW algorithm is advantageous to both of these in terms of CPU times.

Mean CPU time requirements of DICOPT for Models 1 and 2 are 36.61 and 37.58 s, respectively. Minimum and maximum CPU time requirement of DICOPT for both models are 0.01 and 1000 s, respectively. As we mentioned before, CPU time requirement of BARON for Model 1 is 1000 s. CPU requirement of the EFFLOW algorithm is the smallest. It needs 0.01 CPU time on the average. Minimum and maximum CPU time requirement of the algorithm are 0.003 and 0.029 s, respectively.

Our computational study clearly indicates that the proposed EFFLOW algorithm can obtain results that are similar in their solution quality to the results obtained by solving the same problem by using commercial mixed integer nonlinear programming solvers but in a much shorter CPU time. In each instance, we limit the solver running time to 1000 s. One logical question would be measuring the impact of the CPU upper bound on the DICOPT results. The average CPU time for the EFFLOW algorithm was 0.01 s. When we set the CPU time for the DICOPT as 0.1 s, DICOPT could not find a feasible integer solution in nearly all instances and hence it is not possible to make meaningful comparisons. Therefore, we set the CPU upper bound to 1 s for the DICOPT solver. The new results are summarized in Table 6. The numbers of noninteger solutions are increased for both Models 1 and 2 and the mean deviations get slightly worse as expected. When $n=30$, the maximum deviation of Model 1 solved by DICOPT with a time limit of 1 s seems to be greater than when the time limit is 1000 s. This results from a single instance for which no integer feasible solutions could be found with DICOPT in 1 s. Therefore, instead of the maximum deviations, the number of noninteger solutions is a better measure for this comparison. The proposed linearized version, denoted as Model 2, performs better than Model 1 for a shorter CPU time upper bound. In general, the DICOPT solver obtained its best result early on for both models but could not prove its optimality for a long time, especially when $n=30$.

6. Conclusions

In this paper, we studied a nonpreemptive two-machine flowshop environment in which identical jobs are processed. Each job has three operations, one of which is a flexible operation such that the flexible operations should be assigned to one of the machines in order to minimize the makespan. In this study, the processing times of operations are assumed to be controllable, so the processing times of operations should also be determined. We considered a bicriteria objective of minimizing total manufacturing cost and makespan. Since these two objectives cannot be minimized at the same time, we determined a set of efficient discrete points of makespan and manufacturing cost objectives. We proposed two nonlinear mixed integer programs to solve the problem. The models were solved using DICOPT and BARON solvers of GAMS with a time limit of 1000 s. Since the mathematical programming formulations may not be efficient in terms of CPU time, we proposed an algorithm that generates high quality solutions in small CPU time. At each nondominated point, we had to solve two challenging optimization problems to determine the assignment of flexible operations and processing times for each operation simultaneously. We developed some optimality properties of the problem to propose closed form expressions to determine the optimum processing times in a polynomial time for a given assignment of flexible operations. As a future research, this study can be extended by increasing the number of machines and/or flexible operations. Another extension of this paper may be studying different scheduling environments instead of a flowshop.

Acknowledgments

The authors thank to the anonymous referee whose constructive comments improved the quality of the paper significantly. This research is partially supported by the Scientific and Technical Research Council of Turkey under Grant #110M489.

References

[1] Crama Y, Gultekin H. Throughput optimization in two-machine flowshops with flexible operations. Journal of Scheduling 2010;13:227–43.
[2] Gultekin H, Akturk MS, Karasan OE. Bicriteria robotic operation allocation in a flexible manufacturing cell. Computers and Operations Research 2010;37:779–89.
[3] Gupta JND, Koulamas CP, Kyparisis GJ, Potts CN, Strusevich VA. Scheduling three-operation jobs in a two machine flow shop to minimize makespan. Annals of Operations Research 2004;129:171–85.
[4] Janiak A. Minimization of the makespan in a two-machine problem under given resource constraints. European Journal of Operational Research 1998;107:325–37.
[5] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1:61–8.
[6] Karabati S, Kouvelis P. Flow-line scheduling problem with controllable processing times. IIE Transactions 1997;29:1–14.
[7] Kayan RK, Akturk MS. A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. European Journal of Operational Research 2005;167:624–43.
[8] Muth EJ. The reversibility property of production lines. Management Science 1979;25:152–8.
[9] Nowicki E, Zdrzalka S. A two-machine flow shop scheduling problem with controllable job processing times. European Journal of Operational Research 1988;34:208–20.

[10] Patriksson M. A survey on the continuous nonlinear resource allocation problem. European Journal of Operational Research 2008;185:1–46.
[11] Ruiz-Torres AJ, Ablanedo-Rosas JH, Ho JC. Minimizing the number of tardy jobs in the flowshop problem with operations and resource flexibility. Computers and Operations Research 2010;37:282–91.
[12] Shabtay D, Bensoussan Y, Kaspi M. A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. International Journal of Production Economics 2012;136:67–74.
[13] Shabtay D, Kaspi M, Steiner G. The no-wait two-machine flow shop scheduling problem with convex resource-dependent processing times. IIE Transactions 2007;39:539–57.
[14] Shabtay D, Steiner G. A survey of scheduling with controllable processing times. Discrete Applied Mathematics 2007;155(13):1643–66.
[15] T'kindt V, Billaut JC. Multicriteria scheduling: theory, models and algorithms2nd ed. Berlin: Springer; 2006.
[16] Vickson RG. Two single-machine sequencing problems involving controllable job processing times. AIEE Transactions 1980;12:258–62.
[17] Wang J-B, Wang M-Z. Single-machine scheduling to minimize total convex resource consumption with a constraint on total weighted flow time. Computers and Operations Research 2012;39:492–7.
[18] Yedidsion L, Shabtay D, Kaspi M. Complexity analysis of an assignment problem with controllable assignment costs and its applications in scheduling. Discrete Applied Mathematics 2011;159:1264–78.