Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Unrelated parallel machine scheduling with machine processing cost

Hamid Safarzadeh[a] and Seyed Taghi Akhavan Niaki[a*]

[a]*Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11155-9414 Azadi Ave., Tehran 1458889694 Iran*

C H R O N I C L E

Article history:
Received August 10 2022
Received in Revised Format August 31 2022
Accepted October 27 2022
Available online October, 27 2022

Keywords:
Parallel machine scheduling
Machine cost
Green cost
Multiobjective scheduling
Mathematical programming
Pareto optimal front

A B S T R A C T

In practical scheduling problems, some factors such as depreciation cost, green costs like the amount of energy consumption or carbon emission, other resources consumption, raw material cost, etc., are not explicitly related to the machine processing times. Most of these factors can be generally considered as machine costs. Considering the machine cost as another objective alongside the other classical time-driven decision objectives can be an attractive work in scheduling problems. However, this subject has not been discussed thoroughly in the literature for the case the machines have fixed processing costs. This paper investigates a general unrelated parallel machine scheduling problem with the machine processing cost. In this problem, it is assumed that processing a job on a machine incurs a particular cost in addition to processing time. The considered objectives are the makespan and the total cost, which are minimized simultaneously to obtain Pareto optimal solutions. The efficacy of the mathematical programming approach to solve the considered problem is evaluated rigorously in this paper. In this respect, a multiobjective solution procedure is proposed to generate a set of appropriate Pareto solutions for the decision-maker based on the mathematical programming approach. In this procedure, the ϵ-constraint method is first used to convert the bi-objective optimization problem into single-objective problems by transferring the makespan to the set of constraints. Then, the single-objective problems are solved using the CPLEX software. Moreover, some strategies are also used to reduce the solution time of the problem. At the end of the paper, comprehensive numerical experiments are conducted to evaluate the performance of the proposed multiobjective solution procedure. A vast range of problem sizes is selected for the test problems, up to 50 machines and 500 jobs. Furthermore, some rigorous analyses are performed to significantly restrict the patterns of generating processing time and cost parameters for the problem instances. The experimental results demonstrate the mathematical programming solution approach's efficacy in solving the problem. It is observed that even for large-scale problems, a diverse set of uniformly distributed Pareto solutions can be generated in a reasonable time with the gaps from the optimality less than 0.03 most of the time.

1. Introduction

Scheduling, in simple words, is the assignment of resources to activities over time. Scheduling problems are defined in machine environments such as single machines, parallel machines, flow shops, and job shops with different time-driven objectives, mostly related to job completion times (Pinedo, 2016). These objectives include minimizing the *makespan*, minimizing the total completion time, total tardiness minimization, minimizing the number of tardy jobs, etc. In real-world environments, however, there exist some other decision criteria for the scheduling problems such as depreciation cost, the amount of energy consumed, water or other natural resources consumption, the extent of required material, defection/quality cost, etc., that are not explicitly related to the processing times and can be considered as the *machine costs*. Although sometimes the word "cost" is used to name time-driven objectives such as the earliness/tardiness cost, they are different from the intended machine cost, which is not a function of the processing times. Leung et al. (2012) and Lee et al. (2014) pointed out that the time-driven criteria are mostly the customer's objectives because they are interested in their orders' early or on-

* Corresponding author
E-mail: Niaki@Sharif.edu (S. T. A. Niaki)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2023 Growing Science Ltd.
doi: 10.5267/j.ijiec.2022.10.004

time completion times. Meanwhile, Leung et al. (2012) and Lee et al. (2014) mentioned that the machine cost is a type of the service provider's objectives since it expresses production costs. Furthermore, the machine cost is also a useful concept to model sustainability-oriented costs of the production processes, such as energy consumption and carbon emission. From this point of view, the machine cost can be considered a green cost used in establishing green scheduling problems (Safarzadeh & Niaki, 2019).

In the scheduling literature, the machine cost mentioned above can be generally categorized into four classes: 1) machine activation cost, 2) controllable processing cost, 3) fixed processing cost, and 4) time-dependent processing cost. The first class is related to the fixed cost of using a machine. This type of machine cost, in turn, can be classified into two sub-classes. In the first one, the cost is considered for the presence of the machine in the schedule, so the machine selection is also considered as a part of the scheduling problem. However, in the second sub-class, an activation cost is incurred when a machine in the scheduling horizon goes from an idle or turn-off state to an active state to process a job. Both types of the first class of machine cost are addressed several times in the scheduling literature. For the first sub-class, for example, see Dósa and Tan (2010), Xie et al. (2015), Li et al. (2018), Wang and Alidaee (2018), and Kong et al. (2020). Moreover, for the second sub-class, the research works such as Liang et al. (2015), Che et al. (2017), Nasiri et al. (2018), and Meng et al. (2019) can be mentioned.

In the second class of the machine cost, the time of processing a job on a machine can be controlled through its processing cost, where shorter processing times are normally associated with selecting higher processing costs. Many authors such as Ding et al. (2016), Karhi and Shabtay (2018), Wu and Che (2019), Zhang et al. (2020), and Wei et al. (2022) discussed this type of job processing.

In the third class of machine cost, processing a job on a machine is followed by a fixed specific cost .However, the system's overall cost may vary according to the selected schedule, as the jobs can be performed in multiple routes, generally due to the presence of parallel resources with different costs. Indeed, this type of job processing flexibility permits the presence of the processing cost in the problem as a decision factor. Several researchers considered this type of processing cost in the literature (*e.g.*, Leung et al., 2012; Yeh et al., 2015; Mokhtari & Hasani, 2017; Kononov et al., 2019; Hasani & Hosseini, 2020). Nonetheless, the literature on the fixed processing cost, which is also the context of the current study, is significantly scarcer than the machine activation cost and controllable processing cost. However, this type of machine cost may occur in many real cases in which there exist multiple parallel resources with different rates of cost for operation.

The fourth class of machine cost is a newer type discussed in the scheduling literature. In this type of machine cost, the machine's processing cost depends on the time the machine is used; hence, it is usually called the time-of-use (TOU) cost. This type of machine cost is typically utilized in establishing green scheduling problems, including TOU electricity tariffs (*e.g.*, Zeng et al., 2018; Wang et al., 2020; Karimi et al., 2021; Heydar et al., 2021; Pan et al., 2022).

It is evident that when the machine cost criterion is considered in a scheduling problem, a multi-criteria decision-making problem is normally established since at least a time criterion is taken into account in almost each scheduling problem. In some studies, a single decision criterion is held in the objective function, and the others are passed to the constraints set being limited by an upper/lower bound. Nevertheless, in most studies, both the machine cost criterion and a time-driven criterion are present in the problem as the problem aims to set a multiobjective optimization problem.

The parallel machine scheduling problem is the simplest type that embeds the required flexibility for the third type of the machine cost aforementioned, *i.e.,* the fixed processing cost. That is why most of the works that have considered this type of cost are established for this environment. In the parallel machine scheduling problem, there are some single-operation jobs to be processed by several parallel machines. This problem is generally classified into three types: identical, uniform, and unrelated parallel machine scheduling, where identical and uniform are special cases of unrelated environments (Pinedo, 2016). In an unrelated parallel machine scheduling problem, the processing time of a job on a machine is independent of the other processing times. Similarly, it can be assumed in this type of problem that assigning a job to a machine incurs a fixed processing cost that is independent of the other processing costs or times. This enables one to consider the fixed processing cost for the problem as well. The studies addressing the fixed processing cost in a parallel machine scheduling problem are reviewed in detail in the next section.

Mathematical programming is a well-known approach to model and solve scheduling problems. Although most of the scheduling problems are NP-hard, many are solvable using this approach, at least for a considerable segment of practical sizes. Moreover, the efficiency of using mathematical programming to solve scheduling problems is increasing due to improving solution techniques and the development of computer technologies (Ku & Beck, 2016). A significant advantage of mathematical programming is its ability to prove solution optimality. Furthermore, if the problem is not optimally solved, at least an upper bound for the optimal gap of the output solution can be reported by the solver. This value may be small enough such that the resulting solution can be used to approximate the optimal solution. Many researchers have utilized mathematical programming as the major solution approach to analyze scheduling problems in the past two decades. For example, Pan and Chen (2005) examined mixed-integer linear programming (MILP) formulations for a reentrant job shop scheduling problem. Ziaee and Sadjadi (2007) studied a flow shop scheduling problem with different objectives and constraints. They used

mathematical programming to model and solve the problems. Keha et al. (2009) evaluated different types of mathematical models to solve single-machine scheduling problems. Unlu and Mason (2010) compared MILP formulations to solve parallel machine scheduling problems. Naderi et al. (2014) proposed four MILP models for the problem of hybrid flow shop scheduling and compared their performance. Ku and Beck (2016) examined the performance of traditional MILP models for classical job shop scheduling problems. Similar research works have also been conducted for flexible job shop scheduling (Özgüven et al., 2010; Demir & Kürşat İşleyen, 2013) and flexible job shop scheduling with parallel batch processing machines (Ham, 2017). Moreover, Meng et al. (2019) proposed six MILP models for an energy-aware flexible job shop scheduling problem and analyzed their performances through numerical experiments.

This paper addresses an unrelated parallel machine scheduling problem with fixed processing costs. The parallel machines are unrelated, so the time and cost parameters are arbitrary to define a general problem. The considered objectives of the problem are the *makespan* and the total cost, which are minimized simultaneously. This study aims to rigorously analyze the efficacy of the mathematical programming approach to solve the addressed practical problem. In this regard, a multiobjective solution procedure is proposed to extract or estimate a uniformly distributed set of Pareto optimal solutions to the problem in a reasonable time. In this procedure, the constraint method is first utilized to transfer the *makespan* objective into the constraints set. Then, the mathematical programming approach is utilized to solve the induced single-objective problems. To reduce the solution time, two strategies that allow a small gap from the optimality are applied to solve the single-objective as well as to transfer the solutions between successive single-objective problems. The efficiency of the proposed mathematical programming solution procedure in solving the bi-objective problem is demonstrated through a comprehensive set of numerical experiments. The experiments are performed on a broad range of problem sizes up to 50 machines and 500 jobs. Meanwhile, a concrete analysis is carried out in the experiments to significantly simplify and restrict the candidate uniform distributions for generating time and cost parameters.

According to the literature review presented in the next section and to the best of the authors' knowledge, the only works that investigated the described problem in the literature are the ones in Leung et al. (2012), Lee et al. (2014), and Kononov et al. (2019). However, in these works, the structure of the problem objectives and the approach taken to solve the problem are different from the one proposed in the current paper. In other words, none of the above three articles have considered the *makespan* and the total processing cost to be optimized concurrently. Moreover, the performance of the mathematical programming approach has not been evaluated in solving their problems.

The remainder of the paper is outlined as follows. Section 2 presents a related literature review for the considered problem. In Section 3, the problem is defined formally, and its mathematical formulation is given. Section 4 explains the multiobjective solution procedure to obtain the Pareto solutions. Section 5 is devoted to examining the efficacy of the proposed solution approach through comprehensive numerical experiments. Finally, Section 6 concludes the paper, where some directions are suggested for conducting future research.

2. Literature review

The previous section discussed four types of machine costs in the scheduling problems, *i.e.*, machine activation, controllable processing, fixed processing, and time-dependent processing costs. This section reviews the research related to the parallel machine scheduling problem with fixed processing costs, which is the context of this study. Note that the studies in the literature are checked carefully to ignore the ones considering other types of machine costs. That is why some studies on parallel machine scheduling in the literature evaluating machine costs are not mentioned. Moreover, the relation of our research to the reviewed papers is discussed at the end of the section. Several researchers have addressed the fixed processing cost in the parallel machine scheduling problem in the literature. Kolahan & Kayvanfar (2009) analyzed a multiobjective unrelated parallel machine scheduling problem in which a total weighted sum of the machining costs, the earliness-tardiness penalties, and the *makespan* was minimized. They utilized a simulated annealing (SA) algorithm to solve the problem. Leung et al. (2012) addressed a bi-objective scheduling problem in an unrelated parallel machine environment. They assumed that performing a job on a machine is followed by a cost called the machine assignment cost. Their research objectives were to minimize the total cost and either the *makespan* or the total completion time. The authors considered two minimization strategies for the objectives: hierarchically and weighted sum. They analyzed the complexity of the problem for each. Ji et al. (2013) studied a green uniform parallel machine scheduling problem in which each machine has a processing speed and a resource consumption rate. In this problem, the *makespan* is restricted by an upper bound, and the total resource consumption, *i.e.,* the total cost, is minimized as the objective function. The authors proved the NP-hardness of the problem and developed a heuristic method and a particle swarm optimization (PSO) algorithm to solve it. Lee et al. (2014) addressed an unrelated parallel machine scheduling problem with job processing costs. They considered two customer-driven objectives, *i.e.,* the *makespan* and the total completion time, alongside two service-provider-related objectives, *i.e.,* the total cost and the maximum machine cost. They proposed some heuristic algorithms for the two hierarchically-based problems: minimizing the maximum machine cost after minimizing the total completion time and minimizing the total cost after minimizing the *makespan*. Furthermore, Kononov et al. (2019) discussed extensions of the problems considered by Lee et al. (2014) and analyzed some approximation algorithms for the problems addressed. Yeh et al. (2015) investigated a problem similar to the problem in Ji et al. (2013) and compared the performances of three meta-heuristic algorithms, *i.e.*, genetic algorithm (GA),

PSO, and simplified swarm optimization (SSO), by conducting some numerical experiments. Li et al. (2016) considered a type of green parallel machine scheduling problem in which the machines are similar concerning their processing times but have different cost rates, where the total processing cost should not exceed a threshold. They analyzed two problems through heuristic algorithms in which either the *makespan* or the total completion time is minimized. They evaluated the heuristics' performances compared to the one of an exact method in some numerical experiments. Liu et al. (2016) investigated a uniform parallel machine scheduling problem with the possibility of outsourcing. Their work aimed to minimize the total cost, *i.e.,* the sum of the total resource consumption and the total outsourcing cost. They considered an upper bound for the maximum tardiness of the entire job. The authors showed that the preemption case is polynomially solvable. Furthermore, they proposed a branch and bound algorithm to solve the non-preemption case exactly, along with a hybrid meta-heuristic to solve it approximately. Li et al. (2018) considered a uniform parallel machine scheduling problem in non-preemptive and preemptive cases. The problem objective was to minimize the makespan while a given budget limited the total machine cost. They proposed approximation algorithms to analyze the problems. Safarzadeh & Niaki (2019) investigated a uniform parallel machine scheduling problem in green scheduling. They proposed that the machines have specific green cost rates in addition to processing time rates or speeds. Moreover, the problem objective was simultaneously minimizing the makespan and total green cost. The authors applied the ϵ-constraint approach to estimate Pareto optimal solutions through a heuristic approach.

According to the above relevant literature review, one can observe that most of the papers mentioned above address identical or uniform parallel machine environments. However, we consider the unrelated parallel machine environment to analyze a more general problem. In fact, among the papers reviewed in this section, Kolahan and Kayvanfar (2009), Leung et al. (2012), Lee et al. (2014), and Kononov et al. (2019) considered an unrelated parallel machine scheduling problem. Nevertheless, the objective function in the former article differs from our study's. Meanwhile, although the three latter papers address the decision objectives of the *makespan* and total machine cost, which are considered in our problem as well, they utilized the lexicographic or hierarchical approach to optimize them. Moreover, none of the four mentioned papers has examined the performance of the mathematical programming approach to solve the addressed problems.

3. Problem statement and modeling

In this section, the problem is first defined formally. Then, the mathematical formulation of the problem in the form of a mixed-integer linear programming (MILP) model is presented.

3.1. Problem statement

The problem under investigation is an extension of the unrelated parallel machine scheduling problem. In the considered problem, there exist n jobs to be processed on m machines. Each job must be operated by one of these machines without preemption. Besides, each machine affords to process at most one job at a time. All the jobs and the machines are ready at the beginning of the scheduling horizon with no sequence-dependent setup times or costs. Processing a job j on a machine i takes p_{ij} units of time and incurs a cost of c_{ij} units. In addition, there are no predefined relations between the values of time or cost parameters, implying that the machines are unrelated. The question is how to assign the jobs to the machines to minimize the *total cost* (TC) and the completion time of all jobs, *i.e.*, *makespan* (C_{max}), simultaneously.

3.2. Mathematical formulation

Having defined the decision variables as

x_{ij} : A binary variable equals 1 if job j is assigned to machine i; 0 otherwise.

C_{max} : A real variable denoting the *makespan*,

the bi-objective mathematical formulation of the problem is easily derived as

$$min\ C_{max} \tag{1}$$

$$min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2}$$

$subject\ to$

$$\sum_{j=1}^{n} p_{ij}x_{ij} \le C_{max} \quad ; \quad \forall i \tag{3}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad ; \quad \forall j \tag{4}$$

$$C_{max} \ge 0 \ , \ x_{ij} \in \{0,1\} \quad \forall i,j \tag{5}$$

The objectives to minimize the *makespan* and the total cost are stated in Expressions 1 and 2, respectively. Constraint 3 ensures that the *makespan* is greater than or equal to the total processing time of the jobs on any machine. Constraint 4 guarantees that each job is assigned exactly to one machine. Finally, the domains of the decision variables are defined in Constraints 5.

4. Solving the problem

Generally, several approaches have been proposed in the literature to solve multiobjective optimization problems. The most common are *lexicographic ordering, weighted sum*, and *Pareto efficient* methods (Branke et al., 2008). In the *lexicographic ordering* or the *hierarchical optimization* approach, the objectives are first ranked regarding their priorities, and then the problem is solved hierarchically concerning these ranks. The *weighted sum* is another approach in which all the objectives are aggregated with predefined weights to obtain a single objective function. These weights are set based on the objectives' scales and their importance degree. Finding *Pareto efficient* solutions is the last and the most comprehensive approach. A *Pareto solution* is a solution for which the value of an objective function cannot be improved unless the value of at least one of the other objectives is weakened. The Pareto solutions form a set of non-dominated answers allow a trade-off of one objective against the other (Ho et al., 2018). Fig. 1 illustrates the Pareto solutions to a minimization problem in red and green circles. Note that each solution of the two previous approaches is also a Pareto solution. In fact, it is sufficient to check only the Pareto solutions for rational decision-makers who adopt any decision-making approach. Pareto solutions provide diverse choices for the decision-maker with different trade-offs between the problem objectives. The final decision can be made by observing these solutions and regarding the other preferences and conditions (Branke et al., 2008).

There exist two approaches to obtaining the Pareto solutions to an optimization problem: (1) weighting the objectives and (2) the ϵ-*constraint* method (Branke et al., 2008). In the former, the objectives are first aggregated with variable weights. Then, the problem is solved for any specific values of the weights to obtain a Pareto solution for the original problem. Hence, some Pareto solutions can be attained by varying the weights' values and solving the problems. The weakness of this method is that when the Pareto front is non-convex (in a minimization problem), some of the Pareto solutions cannot be identified. For example, in Fig. 1, the red points are not attainable by choosing any values of the weights because they are on a concave part of the Pareto front. In the ϵ-constraint method, however, this shortcoming is eliminated, where each Pareto solution can theoretically be obtained. In this approach, all the objectives but one is transferred to a set of constraints by setting an upper (lower) bound parameter for each if it is minimized (maximized). When specific values are assigned to these bounds, a single-objective optimization problem is specified. On the feasibility condition, its optimal solution is a Pareto solution for the original multiobjective optimization problem. The literature has proven that all of the existent Pareto solutions can be identified by varying the values of the bounds in this approach (Coello et al., 2007).

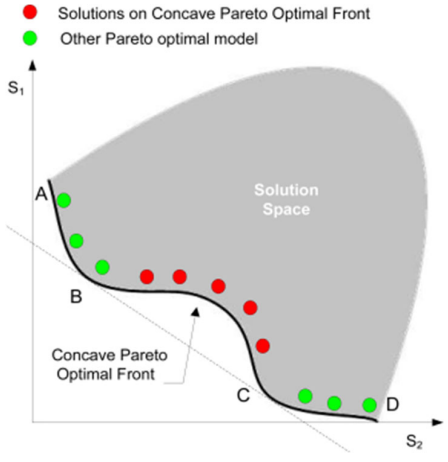

Fig. 1. An example of Pareto solutions on a non-convex Pareto front

For a bi-objective minimization problem with the objective functions f_1 and f_2 at hand, two types of single-objective optimization problems can be generated using the ϵ-constraint method. They are depicted as follows:

$$\begin{cases} \text{Min } f_1 \ , \ \text{Min } f_2 \\ \{ \ Constraints \ Set \ \} \end{cases} \Rightarrow \begin{cases} \text{Min } f_1 \\ f_2 \leq U \\ \{ \ Constraints \ Set \ \} \end{cases} \text{ or } \begin{cases} \text{Min } f_2 \\ f_1 \leq U \\ \{ \ Constraints \ Set \ \} \end{cases}$$

For each type of the above single-objective problem, when the parameter U is varied in a proper domain, all of the Pareto solutions are attainable. It is worth mentioning that in this paper, the term *Pareto* and its related expressions are also used interchangeably for the approximated cases.

4.1. The MILP solution procedure

The ϵ-constraint method is utilized in this paper to obtain or estimate a diverse set of Pareto solutions to the bi-objective minimization problem shown as a MILP model in (1)-(5). The *makespan* objective is transferred to the set of constraints to this aim. After limiting C_{max} by an upper bound T, a single-objective optimization problem is obtained as shown in Expressions (6)-(9). A mathematical programming solver can solve this problem. The only decision variable in this formulation is x_{ij} with the previous definition.

$$min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{6}$$

$$subject\ to$$

$$\sum_{j=1}^{n} p_{ij}x_{ij} \leq T \quad ; \quad \forall i \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad ; \quad \forall j \tag{8}$$

$$x_{ij} \in \{0,1\} \quad ; \quad \forall i,j \tag{9}$$

In this model, no explicit representation of C_{max} is needed, since T in Constraint 7 is equivalent to the considered upper bound for C_{max}. Note that for each specific value of T, a single-objective optimization problem is defined, and on the feasibility condition, its optimal solution will be a Pareto solution for the bi-objective optimization problem at hand. In this case, the total cost objective of the bi-objective model has the same value as the objective value of the single-objective problem. Furthermore, the *makespan* objective that is less than or equal to T can be calculated using the x_{ij} values. As the number of single-objective optimization problems depends on the chosen values of T and hence can be infinite, only a finite number of such problems are solved to obtain a diverse set of Pareto solutions. Accordingly, it is required to generate a vector $\boldsymbol{T}$, which contains proper values of T, to specify the single-objective problems to be solved. To this end, first, the two *extreme problems* attained from the original problem are solved, each of which retains one of the objective functions while ignoring the other one, to acquire the two extreme Pareto solutions. The extreme problems are named E_1 and E_2 whose objective functions are respectively the total cost (TC) and the *makespan* (C_{max}). The values of the *makespan* in the optimal solutions of these problems yield respectively an upper and a lower bound for the $\boldsymbol{T}$ elements, which are called T_{E_1} and T_{E_2} here. Moreover, the single-objective problem corresponding to the ith element of $\boldsymbol{T}$ is called the *intermediate problem* P_i. Problem E_1 is easily solved by assigning each job to the machine with the least corresponding processing cost. To find the optimal solution of E_2, the bi-objective MILP model is solved by ignoring the total cost objective function. As the diagram of the Pareto front is normally like a decreasing convex function (see Figure 2, for instance), the elements in $\boldsymbol{T}$ are selected so that the differences between the successive elements are small at first and then become larger gradually. This helps Pareto solutions to distribute more uniformly along the Pareto front. To this end, the recurrence relation (10), which is suggested based on extensive numerical simulation experiments, is used to generate $\boldsymbol{T}$ with k elements. In this relation, T_0 is initialized by the value of T_{E_2}, based on which the other elements are found recursively. The experimental results obtained in the next section show that this approach of setting the values for T results in affording the uniformity of the Pareto frontier.

$$T_i = \frac{i^2}{\sum_{j=1}^{k+1} j^2}\left(T_{E_1} - T_{E_2}\right) + T_{i-1} \quad ; \quad i = 1, \dots, k+1 \tag{10}$$

4.2. Reducing the solution time

Two strategies are taken in this section to reduce the required time for obtaining Pareto fronts. In the first strategy, the single-objective optimization problems are solved successively so that the output solution of the previous problem can be set as a feasible solution for the next one. Since the elements in $\boldsymbol{T}$ are arranged in an ascending order based on their indices, the feasible region of the P_i problem is a subset of the P_{i+1}'s feasible region, and hence, each solution of the former is feasible for the latter. In addition, the solution of E_2, whose *makespan* is used as a lower bound to generate the elements of $\boldsymbol{T}$, is also feasible for P_1. Accordingly, to apply the first strategy, which is called the *warm-start solution* thereafter, first E_2 and then the P_i problems are solved in the ascending order of their indices to pass the solution of each problem to the next one as a feasible initial solution. The experimental results in the next section demonstrate the significant efficacy of such a solution approach in reducing the total required solution time. As the second strategy to reduce the solution time, a small error is allowed in obtaining the optimal solutions to the single-objective problems. Indeed, the solver of a MILP model, provided that a feasible solution is found, always relates an *optimal gap* defined as $\frac{Best\ Objective\ Value - Lower\ Bound}{Best\ Objective\ Value}$ for the current best solution during a run. This quantity presents an upper bound for the relative gap between the best-obtained solution from the optimality. The solver declares an acceptable optimal gap in solving the problems. If the value of the optimal gap is less than

the declared value, the solution procedure is terminated. In the current research, the considered allowable values for the optimal gaps vary between 0.005 to 0.03 depending on the size and complexity of the problem. Note that this declared gap is an upper bound for the real gap as it may have a lower (or even zero) value. Once again, in the numerical experiments that follow in the next section, it is observed that this negligence technique significantly reduces the required solution time.

5. Computational results

In this section, the performance of the proposed MILP solution procedure is evaluated through extensive numerical experiments. To this end, a broad set of test problems to be solved is generated by a rigorous analysis that restricts and selects the generation patterns. Then, the generated problem instances are solved by defining small allowed optimal gaps, based on which the solution times are examined. We use the ILOG CPLEX 12.6 to solve the MILP problems. In this regard, the models are coded in Visual Studio 2017 via Concert Technology and C#.NET programming. The characteristics of the computer system used to run the experiments are CPU 3.6 GHz Core i3, 8 GB RAM, and operating system of Windows 10. However, as the employed Visual Studio is a 32-bit-based application, at most, 4 GB RAM was most usable for the experiments. Moreover, the required memory became larger than the available RAM several times in some instances. Hence, we used the capabilities of the CPLEX in compressing and transferring a part of the branching tree's data to the node files in the hard disk. We refer the readers to the CPLEX 12.6.0 Manual on how to manage the out-of-memory problem.

5.1. Data generation

A numerical instance to be specified requires determining the number of jobs (n), the number of unrelated parallel machines (m), the processing times (p_{ij}), and the processing costs (c_{ij}). Here, the dimension of a problem is defined in terms of n and m, which vary up to 500 jobs and 50 machines in Table 1.

Table 1
Selected dimensions for the test problems

m	***n***				
	size 1	**size 2**	**size 3**	**size 4**	**size 5**
5	30	100	150	200	300
10	50	100	150	250	300
15	80	130	180	250	300
20	80	100	150	200	300
30	100	150	200	250	350
40	150	200	250	300	400
50	200	250	300	400	500

Continuous uniform probability distributions, common in the literature, are used to generate the processing times and costs. Such a distribution is shown here by $Uni[a \;\; b]$, where a and b are respectively the lower and the upper bounds of the time or the cost. The following two propositions are used to normalize and subsequently restrict the candidate distributions.

Proposition 1. In the solution procedure stated in Section 4, suppose that all the optimal allowable gaps are set to zero, *i.e.*, the single-objective optimization problems are solved to optimality. In this case, multiplying the entire processing times by a fixed positive value α does not change the obtained Pareto solutions. This is also true for the processing costs when multiplied by a positive value β. Moreover, when a real value c is added to all the processing costs, provided that all of them stay non-negative, the Pareto solutions remain unchanged. Note that by a solution, we mean the way of assigning the jobs to the machines, not the values of times or costs that are dependent on the scale of the parameters.

Proof. Multiplying the entire processing times by α clearly holds the extreme solutions unchanged while T_{E_1} and T_{E_2} are scaled by the multiplier α. Subsequently, the $\boldsymbol{T}$ vector is also scaled in the same way according to Relation (10). As in each intermediate problem, the parameter T and all of the processing times are multiplied by α, the optimal solution and the optimal total cost remain unchanged, where the *makespan* is multiplied by α. Hence, the statement of the Proposition for processing times is proven.

If the processing costs are all multiplied by β and then added by c, the extreme solutions and the vector $\boldsymbol{T}$ remain unchanged. Moreover, the makespan and all the time characteristics do not alter in each given problem schedule. However, the total cost is multiplied by β and added by the constant value nc. This implies that in solving each intermediate problem with the total cost objective, the ranking of the feasible solutions is not changed, and consequently, the optimal solution is kept unchanged. Accordingly, all the generated Pareto solutions remain unchanged; hence, the Proposition's statement for the processing costs is also proven. □

Proposition 2. Suppose that some non-negative values are allowed for the optimal gaps in the solution procedure explained in Section 4. Let the relation for the optimal gap is redefined temporarily as $\frac{Best\ Objective\ Value - Optimal\ Objective\ value}{Best\ Objective\ Value}$, *i.e.*, the

lower bound is substituted by the optimal objective value. Then, multiplying all the processing times or the processing costs by a positive constant value does not change the Pareto solutions obtained. However, shifting the processing costs by adding a negative (positive) value c may change the solutions without degrading (upgrading) their quality (a solution has higher quality if it is closer to the Pareto optimal front.)

Proof. Define the *relative objective* of a solution in a single-objective optimization problem as $\frac{Solution\ Objective\ Value - Optimal\ Objective\ Value}{Solution\ Objective\ Value}$. Consider the case of multiplying the time or cost parameters by a specified positive value. Obviously, the extreme problem E_1 is not changed, as it is always solved optimally through the utilized simple assignment. In the extreme problem E_2, the solutions' relative objectives remain unchanged since both the solution objective value and the optimal objective value are scaled similarly. Thus, having a fixed optimal allowable gap, the set of solutions that have relative objectives less than this gap remains unchanged, and hence the output solution of E_2 does not change. Like Proposition 1, the time or cost parameters are scaled in the intermediate problems. Moreover, likewise the extreme problem E_2, the relative objective values of the solutions in the intermediate problems are not modified, subsequently, the solutions do not alter. Accordingly, the statement related to multiplying the time or cost parameters by positive values is proven.

If the cost parameters are added by c, provided that the parameters remain non-negative, then the extreme solution E_2 does not change because this problem does not consider the processing costs. Moreover, in the extreme problem E_1, the ranking of the processing costs is the same as before, so each job is assigned to the former machine and hence the extreme solution remains unchanged. Accordingly, T_{E_1} and T_{E_2}, and subsequently, the $\boldsymbol{T}$ vector is unchanged. However, in the intermediate problems, the objective values of the solutions are added by nc. Therefore, in the relative objective expressions, the numerator is unchanged, while the denominator is added by nc. If c is positive, the values of the relative objectives are reduced. Hence, for a fixed optimal allowable gap, the solutions that have not been accepted previously may now be selected as the output of the solution procedure. If c is negative, the statement is reversed, and the former solutions may now be refused. In conclusion, the solutions may be changed in both cases, but in the former case (when c is positive), the quality of the solutions may be degraded. In contrast, in the latter (when c is negative), the selected solutions may be more qualified. Accordingly, the proof of the Proposition is completed. □

Note that the *lower bound* was substituted by the *optimal objective value* in the definition of the optimal gap to prove Proposition 2 rigorously. However, the utilized proof is also practically valid for the original definition of the optimal gap. This is because while the problem is being solved, the lower bound is very close to the optimal objective value according to the chosen small optimal gaps.

Now the propositions are used to select the uniform distributions to generate time and cost parameters. As the optimal allowable gap is exploited to solve the MILP problems, the results obtained by Proposition 2 are utilized here. However, Proposition 1 can be useful if all the MILP problems are solved to optimality. According to Proposition 2, it is possible to multiply the domain of time or cost distributions by a positive value without affecting the essence of the generated problem instances. This is similar to the case that all the related random generated parameters are multiplied by a positive value, which does not change the resulting Pareto solutions regarding Propositions 2. Thus, a *Uni*[α 100] distribution in which α is a value to be determined is used to generate processing times. Here, three candidate values of 1, 10, and 50 are chosen for α to produce high, moderate, and low variance processing time problems, respectively.

The above argument is also valid for the processing cost distribution; moreover, there is an extra simplification. Suppose the domain of this distribution, and subsequently its generated quantities, are shifted by a constant value such that the lower bound of the domain becomes zero. In this case, the extreme solutions are not changed, and the intermediate Pareto solutions may be improved according to Proposition 2. Therefore, selecting a uniform distribution with the lower bound of zero and an arbitrary positive upper bound is sufficient to generate random processing costs. Here, the *Uni*[0 100] distribution is used to generate processing costs involved in the problem instances.

5.2. Experiments and results

According to the adopted procedure in Section 4, to solve a problem instance, it is required to determine the desired number of Pareto solutions (N), the optimal allowable gap for the extreme problem E_2 (*gap* 1), and the optimal allowable gaps for the intermediate problems P_i (*gap* 2). We set $N = 20$ for all problem instances, considering the extreme solutions. However, the number of resulting solutions may be slightly lower since some single-objective problems may have identical solutions. The optimal allowable gaps are selected in the interval 0.005 to 0.03 to have a reasonable solution time using preliminary experiments. Combining the problems in Table 1 with the three selected values for α, the *problem classes* are defined, based on which ten instances are generated randomly for each class to be solved with the specified optimal gaps.

The results obtained from the main experiment are reported in Tables 2-5, in which each row is devoted to the instances of a problem class. In these tables, $gap1$ and $gap2$ correspond to the considered optimal allowable gaps for solving the extreme problem E_2 and each of the intermediate problems P_i, $i = 1, \dots, 18$, respectively. To control the solution time, a time limit of

600 seconds is set to solve each of the single-objective problems, *i.e.*, the extreme or the intermediate problems. If the run time exceeds this time limit, then the solution of the single-objective problem is terminated, and the time limit is recorded as the solution time. Moreover, $AST1$ and $AST2$ represent the average solution time of the extreme problem and the average total solution time of the entire intermediate problems, respectively. Besides, $ATST$ is the average total solution time of the MILP solution procedure, which is the aggregate of the two previous indicators. Meanwhile, $MST1$ and $MST2$ are respectively the maximum solution time of the extreme problem and the maximum total solution time of the entire intermediate problems among the problem instances. Furthermore, $MTST$ and $SDST$ are respectively the maximum and the standard deviation of total solution time of the MILP solution procedure among the problem instances, and $\bar{N}$ denotes the average number of the resulted Pareto solutions. In these tables, the total number of solution terminations by the time limit in solving the problem instances is reported by $Fail\#$. This indicator is a doublet in which the first element corresponds to the extreme problem, and the second one is for the intermediate problems. Accordingly, the values of these terms are out of 10 and 180, respectively. Similarly, the doublet $FGap$ reports the average optimal gaps of the time-limit-exceeded instances. The first and the second terms correspond respectively to the extreme and the intermediate problems. Additionally, $FMax$ represents the largest index in the vector $\boldsymbol{T}$ that the solution time of its corresponding intermediate problem has exceeded the time limit at least for one instance. As explained later, this indicator shows that in which part of the Pareto front, the solution time mostly exceeds the time limit.

Tables 2-5 show that, in general, the MILP solution procedure has efficiently earned the Pareto solutions with the defined optimal gaps. According to the values of $MTST$, the total solution time is lower than 2 hours for every problem instance, and in most cases, the assigned gaps are attained. However, as denoted by the $Fail\#$ indicator, in some rare single-objective problems, the time limit of 600s is exceeded. However, the resulted gaps are still close to the assigned gaps regarding the values of the $FGap$ indicator. Moreover, concerning the problem types, it is observed that the problems with lower variances of the processing time are harder to solve. Another finding is that increasing the number of jobs in a problem does not necessarily increase the solution time. Accordingly, some critical jobs result in the most challenging problem cases for a certain number of machines. Furthermore, regarding the $FMax$ indicator, it is evident that the first intermediate problems are mostly subject to exceeding the time limits. It will be discussed later that these problems are much harder to solve than the other intermediate problems.

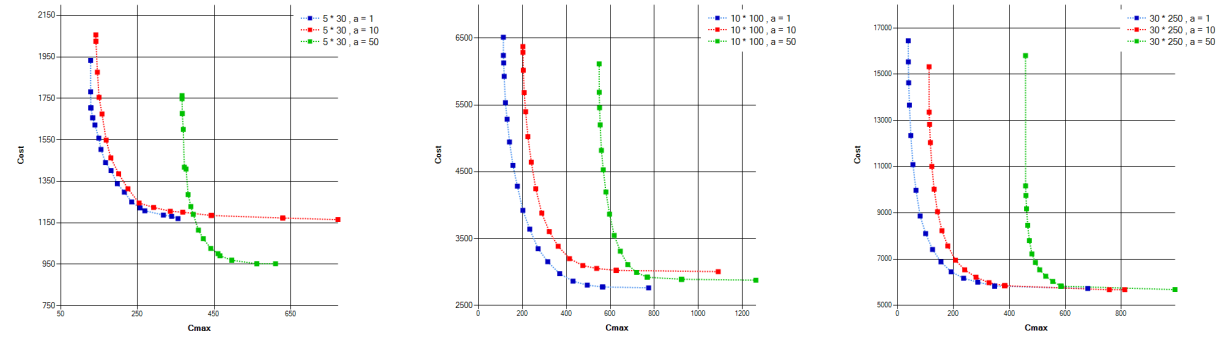

Fig. 2 - Pareto solutions generated for some problem instances

In addition to the above numerical results, the estimated Pareto solutions for some problem instances are displayed in Fig. 2 to give an insight into the solutions attained by the proposed solution procedure. In this figure, the uniformity of the solutions along with the Pareto fronts is evident, implying the effectiveness of using Eq. (10). Some additional experiments are also conducted to evaluate the impact of the solution time reduction strategies. As explained in Section 4.2, the *warm start solution* and the optimal gap declaration are the utilized strategies to solve the problems in shorter times. Accordingly, the MILP solution procedure was applied in some instances of two extra cases, *i.e.,* with smaller optimal gaps and without the warm start approach. The results are presented in Table 6. In this table, five randomly generated instances are solved for each problem class. Some previously introduced indicators are used to compare the three solution strategies, *i.e.,* ordinary, shorter optimal gap, and no warm start solutions. Besides, another indicator ($NoFe\#$) is presented for the latter to show the number of intermediate problems for which a feasible solution cannot be found within the time limit. Note that the indicator $Fail\#$ for the no warm start approach is a single number representing the number of intermediate problems for which the solution time passed the time limit or a feasible solution was not found. In addition, the total solution times of the solution approaches, denoted by $ATST$ in Tables 2-5, are compared. In this regard, the relative total solution times of the two latter approaches with respect to the former are reported by the indicators $RelSG$ and $RelNW$, respectively. Based on these indicators, as well as the total time limit fails ($Total\ Fail\#$), it is evident that both strategies have had remarkable effects in decreasing the solution time. It is concluded that when the gaps are reduced a little in the optimal allowable gap strategy, the solution time of the procedure may be increased much. Moreover, if the warm start solution is not used, even a feasible solution may not be found at all for some intermediate problems. Meanwhile, the solution times of some others may exceed the time limit because of the initial search for a feasible solution or not having an initial qualified solution.

Table 2

Experimental results of applying the MILP solution procedure for the problem instances with 5 and 10 machines

m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$SDST$	$\bar{N}$	$Fail\#$	$FGap$	$Fmax$	m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$SDST$	$\bar{N}$	$Fail\#$	$FGap$	$Fmax$
5	30	1	0.005	0.005	1.5	12.7	14.2	8.3	13.8	19.7	2.2	17.7				10	50	1	0.005	0.005	1.0	14.9	15.9	1.3	17.4	18.5	1.6	19.6			
		10			0.7	13.1	13.8	1.0	15.3	16.3	1.2	17.5						10			1.2	14.7	15.9	2.0	16.5	17.7	1.3	19.2			
		50			0.7	12.4	13.1	0.9	14.2	14.9	1.2	17.7						50			1.1	14.4	15.5	1.7	16.3	17.5	1.3	18.8			
	100	1	0.005	0.005	1.0	15.3	16.4	1.8	16.4	17.1	0.6	19.5					100	1	0.005	0.01	1.6	17.6	19.3	3.9	20.8	23.0	2.2	19.6			
		10			0.9	17.3	18.1	1.1	20.4	21.2	1.6	19.4						10			4.3	25.0	29.3	15.8	48.1	63.9	12.9	19.1			
		50			0.7	32.5	33.2	1.2	62.9	64.2	13.4	18.8						50			1.4	67.2	68.7	2.0	118.4	119.8	26.1	18.1			
	150	1	0.005	0.005	0.9	15.5	16.4	1.6	17.9	18.4	1.3	19.9					150	1	0.005	0.01	2.4	27.0	29.4	4.0	61.0	63.3	13.1	19.5			
		10			0.7	17.1	17.8	1.0	20.9	21.7	2.5	19.7						10			7.3	47.2	54.5	14.9	82.6	90.7	18.1	19.3			
		50			0.5	82.3	82.8	1.1	167.5	168.0	42.0	19.1						50			61.5	462.6	524.1	600.1	1149.9	1312.7	471.9	17.9			
	200	1	0.005	0.01	0.6	9.8	10.4	1.6	11.4	12.5	1.2	19.7					250	1	0.005	0.01	4.9	46.8	51.7	15.0	75.8	79.3	18.4	19.6			
		10			0.5	10.7	11.2	0.8	12.9	13.7	1.4	19.6						10			3.4	231.6	235.0	8.4	675.2	678.2	219.0	19.4			
		50			0.4	25.8	26.2	0.6	71.3	71.7	16.9	18.0						50			2.9	1005.0	1007.8	9.6	2478.0	2481.2	646.1	17.9	(0,2)	(-,0.012)	2
	300	1	0.005	0.01	0.5	8.4	8.9	0.8	9.4	10.0	1.0	19.8					300	1	0.005	0.01	2.0	56.9	58.9	6.1	109.8	111.3	33.6	19.5			
		10			0.4	8.5	8.9	0.5	10.0	10.4	0.9	19.5						10			1.6	315.1	316.8	3.3	692.2	694.8	199.0	19.4	(0,1)	(-,0.012)	1
		50			0.4	10.5	10.9	0.4	12.5	13.0	1.4	18.3						50			1.9	1134.4	1136.3	3.6	2120.5	2124.0	726.8	17.6	(0,6)	(-,0.013)	3

Table 3

Experimental results of applying the MILP solution procedure for the problem instances with 15 and 20 machines

m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$SDST$	$\bar{N}$	$Fail\#$	$FGap$	$FMax$	m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$SDST$	$\bar{N}$	$Fail\#$	$FGap$	$FMax$
15	80	1	0.005	0.01	1.9	13.6	15.4	3.1	15.1	17.5	1.3	19.8				20	80	1	0.005	0.01	1.9	11.5	13.3	3.8	13.0	16.8	1.6	19.9			
		10			3.6	16.3	19.8	8.4	20.2	28.5	3.6	19.0						10			3.0	14.6	17.6	4.2	18.2	21.0	1.8	19.1			
		50			26.4	45.7	72.0	198.0	76.8	224.0	55.3	18.8						50			5.9	17.2	23.0	15.8	21.4	37.1	5.8	19.2			
	80	10	0.005	0.01	26.4	45.7	72.0	0.0	0.0	4.9	986.1	18.8					100	1	0.005	0.01	3.6	14.1	17.8	9.7	16.7	23.8	3.5	19.6			
		10			26.7	33.3	60.0	62.0	62.0	118.7	31.8	18.4						10			16.2	23.1	39.2	28.4	29.3	50.7	10.9	19.5			
		50			600.2	483.8	1084.0	600.9	1490.8	2091.0	407.7	17.0	(10,0)	(0.011,-)				50			9.2	32.3	41.6	15.4	45.6	60.3	10.4	18.8			
	180	1	0.005	0.02	23.3	23.1	46.5	34.9	47.7	82.6	17.7	18.7					150	1	0.01	0.02	26.0	21.8	47.8	56.7	54.3	98.8	29.2	18.9			
		10			297.8	168.2	465.9	600.0	402.6	885.5	273.3	18.6						10			154.3	120.9	275.2	600.1	389.1	989.2	320.0	18.6	(1,0)	(0.012,-)	
		50			4.1	230.5	234.7	16.1	740.3	742.9	191.4	17.7						50			481.2	911.1	1392.3	600.2	1879.0	2479.1	602.9	17.4	(8,2)	(0.024,0.025)	4
	250	1	0.005	0.02	270.8	70.7	341.5	600.1	305.4	905.4	308.2	18.9	(1,0)	(0.006,-)			200	1	0.01	0.02	64.5	32.9	97.4	268.1	77.6	293.6	75.5	18.5			
		10			237.9	609.1	846.9	600.1	1007.7	1534.9	393.2	18.5						10			90.7	214.1	304.8	328.5	390.6	719.1	177.5	18.5			
		50			185.5	2120.5	2306.0	600.2	3471.3	4071.5	943.9	17.1	(0,11)	(-,0.033)	5			50			2.9	368.7	371.5	4.6	756.0	760.1	178.5	16.8			
	300	1	0.005	0.02	310.4	113.4	423.9	600.1	349.7	949.8	334.1	18.9					300	1	0.01	0.02	47.7	112.5	160.2	136.9	215.2	283.4	92.0	19.0			
		10			122.7	467.5	590.2	600.1	763.7	1306.6	360.1	18.6	(0,4)	(-,0.023)	1			10			14.5	777.7	792.2	30.6	1267.1	1296.8	379.0	18.5	(0,7)	(-,0.024)	2
		50			123.7	376.6	500.4	600.2	823.4	1064.2	340.4	17.3	(1,0)	(0.006,-)				50			2.4	535.0	537.4	5.2	992.5	996.1	216.2	16.8			

Table 4
Experimental results of applying the MILP solution procedure for the problem instances with 30 and 40 machines

m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$VTST$	$\bar{N}$	$Fail\#$	$FGap$	$FMax$	m	n	α	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$VTST$	$\bar{N}$	$Fail\#$	$FGap$	$FMax$
30	100	1	0.01	0.02	2.4	9.4	11.8	5.9	10.3	15.9	2.2	19.2				40	150	1	0.02	0.03	16.7	10.1	26.8	85.0	10.9	95.9	27.8	18.9			
		10			5.1	12.2	17.3	12.6	14.0	24.6	2.7	18.8						10			31.5	22.9	54.4	75.2	31.0	99.8	29.6	18.5			
		50			5.2	28.4	33.5	8.8	63.0	66.7	12.8	17.5						50			188.3	313.6	501.9	600.2	565.6	1165.8	309.8	17.7	(2,0)	(0.029,-)	
	150	1	0.02	0.02	16.2	15.3	31.6	51.0	18.0	68.3	16.6	18.8					200	1	0.02	0.03	93.2	22.4	115.6	260.3	36.5	287.7	72.1	18.7			
		10			60.5	64.5	125.0	139.1	186.7	281.0	79.6	18.9						10			157.1	85.7	242.8	600.2	141.6	741.8	245.7	18.0	(2,0)	(0.028,-)	
		50			1.4	59.7	61.1	5.0	99.6	101.2	22.0	18.2						50			3.1	129.9	133.0	12.9	300.5	305.9	83.7	17.5			
	200	1	0.02	0.03	51.0	20.9	71.9	110.7	48.2	132.1	32.0	18.6					250	1	0.02	0.03	230.1	55.6	285.7	600.2	172.6	772.8	299.9	18.7	(3,0)	(0.025,-)	
		10			56.1	161.2	217.3	214.3	320.8	535.2	132.8	18.1						10			268.7	686.4	955.1	600.2	1209.3	1679.2	492.5	17.7	(1,3)	(0.024,0.048)	1
		50			332.2	2280.6	2612.8	600.2	3274.1	3874.3	735.2	16.4	(4,18)	(0.024,0.043)	6			50			205.7	1405.9	1611.7	601.0	1982.7	2229.2	453.9	16.9	(3,1)	(0.037,0.033)	5
	250	1	0.02	0.03	101.6	52.2	153.9	506.7	118.2	557.9	144.0	18.5					300	1	0.03	0.03	21.6	73.5	95.1	75.2	142.5	217.8	57.2	18.3			
		10			24.2	408.9	433.2	49.1	761.8	773.4	243.5	18.4						10			20.5	855.2	875.7	38.2	1234.6	1266.2	225.4	17.9			
		50			428.6	2043.3	2471.9	600.3	2767.8	3368.1	629.1	16.1	(7,12)	(0.037,0.041)	6			50			540.8	4460.2	5001.0	601.0	5027.3	5628.3	366.2	16.3	(8,60)	(0.036,0.05)	8
	350	1	0.02	0.03	48.7	80.8	129.5	307.3	206.1	358.4	109.7	18.3					400	1	0.03	0.03	6.9	151.7	158.5	9.4	254.8	262.1	66.1	18.3			
		10			12.6	688.8	701.4	24.5	1225.0	1233.6	347.8	18.0	(0,3)	(-,0.036)	1			10			8.7	1359.0	1367.6	13.9	1860.0	1864.6	294.4	17.7	(0,2)	(-,0.033)	2
		50			42.9	4113.2	4156.1	152.5	5100.8	5124.9	514.2	16.2	(0,53)	(-,0.058)	6			50			0.5	1389.2	1389.7	0.6	1877.6	1878.0	245.6	15.9			

Table 5
Experimental results of applying the MILP solution procedure for the problem instances with 50 machines

m	n	$alpha$	$gap1$	$gap2$	$AST1$	$AST2$	$ATST$	$MST1$	$MST2$	$MTST$	$SDST$	$\bar{N}$	$Fail\#$	$FGap$	$FMax$
50	200	1	0.02	0.03	153.7	15.0	168.7	600.2	20.9	619.1	236.7	18.5	(2,0)	(0.026,-)	
		10			22.0	53.2	75.2	109.3	123.0	174.7	47.8	18.5			
		50			63.0	86.1	149.0	600.2	151.9	691.5	193.0	16.9	(1,0)	(0.028,-)	
	250	1	0.03	0.03	234.7	30.5	265.3	600.4	76.7	677.0	264.7	18.1	(2,0)	(0.033,-)	
		10			30.1	222.9	252.9	87.2	358.5	409.7	93.9	18.2			
		50			0.5	333.4	333.9	1.2	608.2	608.6	156.0	16.6			
	300	1	0.03	0.03	313.8	102.7	416.5	600.2	202.6	732.3	247.2	18.2	(1,0)	(0.031,-)	
		10			100.7	858.0	958.7	315.9	1725.3	1743.2	392.3	17.9	(0,1)	(-,0.036)	3
		50			0.5	915.0	915.5	0.7	1126.9	1127.3	175.9	16.0			
	400	1	0.03	0.03	49.5	123.2	172.7	159.6	249.4	333.6	84.1	18.1			
		10			39.2	1799.6	1838.8	69.6	2595.1	2638.1	365.5	17.9	(0,7)	(-,0.037)	3
		50			0.7	1889.5	1890.2	2.0	2548.3	2548.9	322.4	16.2	(0,1)	(-,0.031)	9
	500	1	0.03	0.03	13.7	198.9	212.6	35.6	313.5	327.8	62.5	18.4			
		10			19.4	2008.0	2027.5	23.1	2330.9	2351.6	328.5	17.6	(0,15)	(-,0.039)	3
		50			0.9	2862.5	2863.3	2.7	3479.7	3480.4	418.7	16.0	(0,5)	(-,0.036)	9

Table 6
Results for the evaluation of the solution reduction time strategies in the MILP solution approach

m	n	α	Ordinary Solution					Shorter Gap Solution					No Warm Start			Rel. Solution Time	
			gap 1	*gap 2*	*AST1*	*AST2*	*Fail#*	*gap 1*	*gap 2*	*AST1*	*AST2*	*Fail#*	*AST2*	*NoFe#*	*Fail#*	*RelSG*	*RelNW*
5	100	1	0.005	0.005	1.03	13.12		0	0	2.29	17.45		4.22			1.4	0.4
		10			0.81	14.57				1.07	21.58		7.50			1.5	0.5
		50			0.82	28.46				6.83	48.57		269.64	2	2	1.9	9.2
10	150	1	0.005	0.01	6.71	17.93		0	0.005	6.51	28.23		382.41	1	3	1.4	15.8
		10			13.68	61.03				18.88	64.42		1307.00	8	10	1.1	17.7
		50			1.50	234.92				350.68	530.27	(1,0)	1951.79	14	14	3.7	8.3
15	250	1	0.005	0.02	497.04	40.91	(1,0)	0	0.01	418.24	223.89	(3,0)	966.79	7	7	1.2	2.7
		10			282.66	192.15	(1,0)			600.07	661.71	(5,2)	1934.02	15	15	2.7	4.7
		50			76.78	1584.73	(0,6)			600.15	2467.66	(5,11)	3352.08	22	22	1.8	2.1
20	150	1	0.01	0.02	16.51	12.03		0.005	0.01	15.75	18.59		253.33	2	2	1.2	9.5
		10			139.26	45.33				248.13	108.24	(1,0)	1115.94	9	9	1.9	6.8
		50			600.12	579.08	(5,1)			373.69	1158.52	(3,1)	2330.25	11	15	1.3	2.5
30	200	1	0.02	0.03	28.58	12.98		0.01	0.02	323.02	35.37	(1,0)	738.47	6	6	8.6	18.5
		10			61.93	119.94				600.17	361.95	(4,1)	1267.96	10	10	5.3	7.3
		50			367.42	1752.84	(3,7)			600.21	2414.07	(5,15)	3610.80	16	22	1.4	1.9
40	300	1	0.03	0.03	30.68	44.61		0.02	0.02	393.39	100.06	(3,0)	1226.19	10	10	6.6	16.7
		10			11.90	666.31	(0,2)			105.05	1522.10	(0,7)	1373.51	8	8	2.4	2.0
		50			441.84	2741.76	(3,10)			485.42	3400.58	(4,16)	4588.49	15	30	1.2	1.6
50	400	1	0.03	0.03	126.95	65.65		0.02	0.02	600.34	255.44	(5,1)	1135.00	9	9	4.4	6.6
		10			25.32	938.25				207.08	1759.89	(0,6)	2256.44	8	10	2.0	2.4
		50			0.53	317.34				0.80	977.65		3585.28	13	14	3.1	11.3
Total Fail#							(13,26)					(40,60)			218		

Table 7
Detailed average solution times of the single-objective problems in some problem classes

m	n	α	Extreme Problem	Intermediate Problems																
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	100	1	1.09	1.06	0.63	0.92	0.73	0.74	0.60	0.69	0.64	0.43	0.48	0.41	0.34	0.42	0.40	0.26	0.28	0.32	0.25
		10	0.57	0.98	1.05	0.89	0.70	0.75	0.81	0.57	0.79	0.58	0.46	0.51	0.45	0.41	0.37	0.40	0.29	0.28	0.32
		50	0.39	3.08	1.81	1.98	1.32	1.10	1.45	0.93	0.91	0.64	0.55	0.68	0.50	0.79	0.31	0.42	0.22	0.23	0.37
10	150	1	1.28	2.47	1.37	1.16	0.74	0.77	0.66	0.56	0.55	0.52	0.48	0.49	0.44	0.41	0.33	0.39	0.28	0.45	0.28
		10	1.16	6.11	9.76	1.77	1.80	1.06	0.75	0.73	0.67	0.55	0.51	0.45	0.66	0.36	0.41	0.26	0.32	0.28	0.25
		50	1.19	5.91	19.76	10.71	4.46	7.14	2.61	2.84	1.42	2.97	1.58	1.18	0.48	0.44	0.38	0.30	0.28	0.24	0.29
15	250	1	1.12	2.77	0.94	0.62	0.42	0.41	0.35	0.30	0.34	0.35	0.36	0.28	0.29	0.37	0.33	0.34	0.22	0.26	0.34
		10	0.74	11.67	1.51	1.79	1.91	1.00	0.88	0.39	0.38	0.36	0.33	0.41	0.37	0.38	0.34	0.47	0.29	0.22	0.31
		50	0.53	65.00	37.00	8.30	12.52	12.01	8.71	6.31	5.99	3.24	0.55	0.41	0.40	0.29	0.29	0.29	0.25	0.30	0.28
20	150	1	6.43	2.87	0.81	0.83	0.78	0.71	0.56	0.58	0.38	0.44	0.42	0.40	0.31	0.45	0.35	0.32	0.24	0.26	0.32
		10	13.57	21.79	11.54	3.36	2.21	1.78	1.65	1.28	0.96	0.57	0.48	0.40	0.41	0.38	0.32	0.25	0.29	0.61	0.28
		50	366.44	5.95	15.19	22.29	28.31	31.08	15.75	10.48	4.22	2.22	0.44	0.35	0.32	0.28	0.26	0.31	0.26	0.25	0.31
30	200	1	13.79	5.41	2.41	0.94	0.51	0.52	0.43	0.50	0.50	0.38	0.38	0.34	0.32	0.30	0.30	0.27	0.29	0.48	0.22
		10	12.97	21.45	4.47	2.34	7.22	1.18	1.13	0.67	0.51	0.39	0.42	0.39	0.89	0.30	0.28	0.28	0.30	0.29	0.24
		50	15.78	297.25	237.68	224.11	123.96	81.99	27.66	9.19	1.22	1.18	0.39	0.30	0.27	0.27	0.29	0.27	0.28	0.32	0.26
40	300	1	30.14	22.52	1.65	0.69	0.61	0.55	0.45	0.39	0.42	0.36	0.35	0.31	0.35	0.30	0.27	0.31	0.27	0.25	0.30
		10	21.46	279.08	101.56	59.23	14.50	2.23	1.44	1.02	0.62	0.45	0.41	0.39	0.42	0.28	0.27	0.46	0.24	0.33	0.25
		50	600.83	357.99	542.02	573.21	432.23	203.17	67.54	6.08	1.11	0.55	0.50	0.32	0.28	0.47	0.25	0.30	0.27	0.29	0.31
50	400	1	55.74	68.40	3.69	1.08	0.79	0.79	0.46	0.52	0.56	0.58	0.49	0.47	0.35	0.28	0.34	0.27	0.30	0.34	0.26
		10	24.29	192.86	160.83	47.16	36.83	3.26	1.44	1.04	0.65	0.65	0.59	0.62	0.40	0.27	0.52	0.26	0.32	0.30	0.33
		50	0.54	68.26	67.16	20.82	3.88	2.74	1.76	1.36	1.44	1.33	1.12	0.38	0.36	0.30	0.30	0.31	0.32	0.30	0.29

Another finding observed in the experiments is the dissimilarity of solution times of the intermediate problems. In fact, the first intermediate problems are much more challenging to solve than the others. The average solution times of the intermediate problems for some problem classes are reported in Table 7 to illustrate this fact. Again, five randomly generated instances are solved from each selected problem class, and each row is devoted to a problem class. In addition, the average solution times of the intermediate problems in some problem classes in Table 7 are plotted in two charts in Fig. 3.

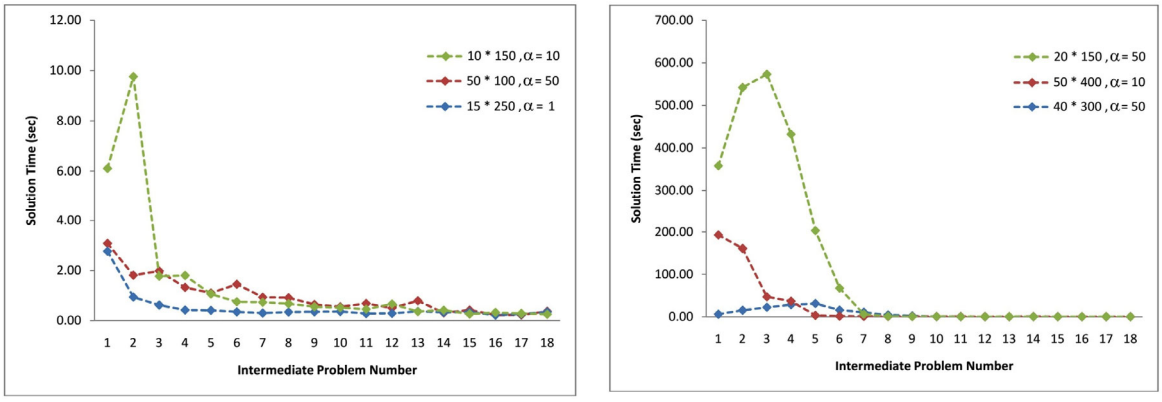

Fig. 3. The average solution times of the intermediate problems in Table 7 for some problem classes

Observing Table 7 and Fig. 3, it is evident that the solution times of the first intermediate problems are much longer than those of the others. However, in some problem classes, the solution times are not monotonically decreasing for the first several intermediate problems. Probably, it is due to the injection of the extreme solution to the first intermediate problem. Indeed, this job provided a handy solution for the problem, causing the actual solution time to be reduced. It seems that this effect has been transmitted to several consequent intermediate problems in some cases, mainly because these problems are close to each other in terms of the values of T generated using Eq. (10).

6. Conclusion and recommendations for future research

In this paper, an unrelated parallel machine scheduling problem with machine processing costs was investigated. Simultaneous minimization of the makespan and the total cost was aimed at this problem. In this respect, a solution procedure based on the mathematical programming approach was developed to obtain a diverse, uniformly distributed set of Pareto optimal or near-optimal solutions. Such a set of solutions provides valuable choices for the decision-maker to select the final solution of the problem based on the actual conditions and preferences. In the developed solution procedure, the ϵ-constraint method was first used to transfer the *makespan* objective into the constraints set. Then, the induced single-objective problems were solved in a structured way using a MILP solver to generate the desired set of Pareto solutions, called the MILP solution procedure. While obtaining the optimal solution was not necessarily aimed, an accurate approximation of the optimal solution was found employing this approach. Moreover, the single-objective problems were solved sequentially, for which the solution of the previous problem was used as a starting solution to the next problem. These two strategies were used in the MILP solution procedure to reduce the total solution time.

The efficacy of the MILP solution procedure was assessed through extensive computational experiments. To this end, test problems were generated with a broad range of dimensions, up to 50 machines and 500 jobs. Furthermore, two propositions were proven rigorously to analyze the candidate uniform probability distributions in generating the processing time and cost parameters. Using the propositions, it was concluded that the only parameter required to be determined to generate processing time and cost parameters was α in $Uni[\alpha\ \ 100]$ (the corresponding uniform distribution for processing times). Consequently, three distinct values were selected for this parameter to have different types of problem instances.

The generated problem instances were solved using the CPLEX software through the proposed MILP solution procedure. A time limit of 600s and an acceptable optimal gap between 0.005 and 0.03 were set to solve each single-objective problem. The solution to a problem was terminated when either of the above two conditions was met. The results showed that the mathematical programming approach was significantly efficient for the studied problem since most of the single-objective problems were solved by the assigned optimal gap within the time limit. Moreover, in rare cases where the solution time exceeded the time limit, the average optimal gaps were still close to the intended optimal gaps. Furthermore, it was observed that the adopted multiobjective solution procedure results in a diverse and uniformly distributed set of Pareto solutions to the problem instances. Another finding was that obtaining a Pareto solution close to the makespan minimization part of the Pareto front was much harder to find than the others. In summary, it was concluded that mathematical programming is an efficient

approach to estimating the Pareto front of the bi-objective optimization problem through the proposed solution procedure, even for large-scale problem instances.

For future studies, developing other multiobjective solution approaches such as using evolutionary algorithms and comparing their performance to the one of the current paper is suggested. Moreover, some different extra assumptions such as batch processing and the existence of sequence-dependent setup times and costs involved in scheduling problems can be taken into account for this problem. Fixed processing costs can also be investigated for the scheduling problems in more sophisticated flexible environments such as hybrid flow shops or flexible job shops.

References

Branke, J., Branke, J., Deb, K., Miettinen, K., & Slowiński, R. (Eds.). (2008). Multiobjective optimization: Interactive and evolutionary approaches (Vol. 5252). Springer Science & Business Media.

Che, A., Wu, X., Peng, J., & Yan, P. (2017). Energy-efficient bi-objective single-machine scheduling with power-down mechanism. *Computers & Operations Research, 85,* 172-183.

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). Evolutionary algorithms for solving multi-objective problems (Vol. 5, pp. 79-104). New York: Springer.

CPLEX 12.6.0 Manual, ILOG Reference of "Running out of memory troubleshooting" - Retrieved from < https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/troubleshoot/61_mem_gone.html >.

Demir, Y., & İşleyen, S. K. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling, 37*(3), 977-988.

Ding, J. Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*(3), 758-771.

Dósa, G., & Tan, Z. (2010). New upper and lower bounds for online scheduling with machine cost. *Discrete Optimization, 7*(3), 125-135.

Ham, A. (2017). Flexible job shop scheduling problem for parallel batch processing machine with compatible job families. *Applied Mathematical Modelling, 45*, 551-562.

Hasani, A., & Hosseini, S. M. H. (2020). A bi-objective flexible flow shop scheduling problem with machine-dependent processing stages: Trade-off between production costs and energy consumption. *Applied Mathematics and Computation*, *386*, 125533.

Heydar, M., Mardaneh, E., & Loxton, R. (2022). Approximate dynamic programming for an energy-efficient parallel machine scheduling problem. *European Journal of Operational Research, 302*(1), 363-380.

Ho, W. H., Chiu, Y. H., & Chen, Y. J. (2018). Multi-objective Pareto adaptive algorithm for capacitated lot-sizing problems in glass lens production. *Applied Mathematical Modelling, 53*, 731-738.

Ji, M., Wang, J. Y., & Lee, W. C. (2013). Minimizing resource consumption on uniform parallel machines with a bound on makespan. *Computers & Operations Research, 40*(12), 2970-2974.

Karhi, S., & Shabtay, D. (2018). Single machine scheduling to minimise resource consumption cost with a bound on scheduling plus due date assignment penalties. *International Journal of Production Research, 56*(9), 3080-3096.

Karimi, S., Kwon, S., & Ning, F. (2021). Energy-aware production scheduling for additive manufacturing. *Journal of Cleaner Production, 278*, 123183.

Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering, 56*(1), 357-367.

Kolahan, F., & Kayvanfar, V. (2009). A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines. *International Journal of Industrial and Manufacturing Engineering, 3*(11), 1406-1409.

Kong, M., Pei, J., Liu, X., Lai, P. C., & Pardalos, P. M. (2020). Green manufacturing: Order acceptance and scheduling subject to the budgets of energy consumption and machine launch. *Journal of Cleaner Production, 248*, 119300.

Kononov, A. V., Kovalyov, M. Y., & Lin, B. M. (2019). Minimizing machine assignment costs over Δ-approximate solutions of the scheduling problem P|| Cmax. *Theoretical Computer Science, 793,* 70-78.

Ku, W. Y., & Beck, J. C. (2016). Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research, 73*, 165-173.

Lee, K., Leung, J. Y., Jia, Z. H., Li, W., Pinedo, M. L., & Lin, B. M. (2014). Fast approximation algorithms for bi-criteria scheduling with machine assignment costs. *European Journal of Operational Research, 238*(1), 54-64.

Leung, J. Y. T., Lee, K., & Pinedo, M. L. (2012). Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics, 139*(1), 321-329.

Li, K., Zhang, H. J., Cheng, B. Y., & Pardalos, P. M. (2018). Uniform parallel machine scheduling problems with fixed machine cost. *Optimization Letters, 12*(1), 73-86.

Li, K., Zhang, X., Leung, J. Y. T., & Yang, S. L. (2016). Parallel machine scheduling problems in green manufacturing industry. *Journal of Manufacturing Systems, 38*, 98-106.

Liang, P., Yang, H. D., Liu, G. S., & Guo, J. H. (2015). An ant optimization model for unrelated parallel machine scheduling with energy consumption and total tardiness. *Mathematical Problems in Engineering, e907034*.

Liu, Z., Lee, W. C., & Wang, J. Y. (2016). Resource consumption minimization with a constraint of maximum tardiness on parallel machines. *Computers & Industrial Engineering, 97*, 191-201.

Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019). MILP models for energy-aware flexible job shop scheduling problem. *Journal of Cleaner Production, 210*, 710-723.

Mokhtari, H., & Hasani, A. (2017). An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers & Chemical Engineering, 104*, 339-352.

Naderi, B., Gohari, S., & Yazdani, M. (2014). Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling, 38*(24), 5767-5780.

Nasiri, M. M., Abdollahi, M., Rahbari, A., Salmanzadeh, N., & Salesi, S. (2018). Minimizing the energy consumption and the total weighted tardiness for the flexible flowshop using NSGA-II and NRGA. *Journal of Industrial and Systems Engineering, 11*(Special issue: 14th International Industrial Engineering Conference), 150-162.

Özgüven, C., Özbakır, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling, 34*(6), 1539-1548.

Pan, J. C. H., & Chen, J. S. (2005). Mixed binary integer programming formulations for the reentrant job shop scheduling problem. *Computers & Operations Research, 32*(5), 1197-1212.

Pan, R., Wang, Q., Li, Z., Cao, J., & Zhang, Y. (2022). Steelmaking-continuous casting scheduling problem with multi-position refining furnaces under time-of-use tariffs. *Annals of Operations Research, 310*(1), 119-151.

Pinedo, M.L. (2016). *Scheduling: Theory, Algorithms, and Systems.* 5th ed. 2016 edition. ed. Springer, New York.

Safarzadeh, H., & Niaki, S. T. A. (2019). Bi-objective green scheduling in uniform parallel machine environments. *Journal of Cleaner Production, 217*, 559-572.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering, 58*(4), 785-800.

Wang, H., & Alidaee, B. (2018). Unrelated parallel machine selection and job scheduling with the objective of minimizing total workload and machine fixed costs. *IEEE Transactions on Automation Science and Engineering, 15*(4), 1955-1963.

Wang, S., Wang, X., Chu, F., & Yu, J. (2020). An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *International Journal of Production Research, 58*(8), 2283-2314.

Wei, Z., Liao, W., & Zhang, L. (2022). Hybrid energy-efficient scheduling measures for flexible job-shop problem with variable machining speeds. *Expert Systems with Applications, 197*, 116785.

Wu, X., & Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega, 82*, 155-165.

Xie, F., Xu, Z., Zhang, Y., & Bai, Q. (2015). Scheduling games on uniform machines with activation cost. *Theoretical Computer Science, 580*, 28-35.

Yeh, W. C., Chuang, M. C., & Lee, W. C. (2015). Uniform parallel machine scheduling with resource consumption constraint. *Applied Mathematical Modelling, 39*(8), 2131-2138.

Zeng, Y., Che, A., & Wu, X. (2018). Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization, 50*(1), 19-36.

Zhang, L., Deng, Q., Gong, G., & Han, W. (2020). A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption. *International Journal of Production Research, 58*(22), 6826-6845.

Ziaee, M., & Sadjadi, S. J. (2007). Mixed binary integer programming formulations for the flow shop scheduling problems. A case study: ISD projects scheduling. *Applied Mathematics and Computation, 185*(1), 218-228.