RAIRO-Oper. Res. 55 (2021) S1447–S1467
https://doi.org/10.1051/ro/2020044

RAIRO Operations Research
www.rairo-ro.org

MULTI-OBJECTIVE MULTI-FACTORY SCHEDULING

Javad Behnamian[1] and Seyyed Mohammad Taghi Fatemi Ghomi[2,*]

Abstract. This paper introduces a multi-factory scheduling problem with heterogeneous factories and parallel machines. This problem, as a major part of supply chain planning, includes the finding of a suitable factory for each job and the scheduling of the assigned jobs at each factory, simultaneously. For the first time, this paper studies multi-objective scheduling in the production network in which each factory has its customers and demands can be satisfied by itself or other factories. In other words, this paper assumes that jobs can transfer from the overloaded machine in the origin factory to the factory, which has fewer workloads by imposing some transportation times. For simultaneous minimization of the sum of the earliness and tardiness of jobs and total completion time, after modeling the scheduling problem as a mixed-integer linear program, the existing multi-objective techniques are analyzed and a new one is applied to our problem. Since this problem is NP-hard, a heuristic algorithm is also proposed to generate a set of Pareto optimal solutions. Also, the algorithms are proposed to improve and cover the Pareto front. Computational experiences of the heuristic algorithm and the output of the model implemented by CPLEX over a set of randomly generated test problems are reported.

Mathematics Subject Classification. 90B35, 68M14, 90C29, 90C59.

Received January 5, 2019. Accepted April 28, 2020.

1. Introduction

In the context of supply chains as a major part of the globalization trend, the jobs can be viewed as various intermediate products that are supplied from an upstream supplier and need to be further processed in one of several factories before their delivery dates to a downstream buyer. Since the supply chain (SC) can consist of multiple layers, including suppliers, manufacturers, distributors, retail outlets, and consumers, its planning and scheduling can be extremely complicated.

Today, scheduling problems have been widely studied by researchers. Among these studies, the majority of research focuses on single factory planning. But, since large workload requirements are regular in some cases, multi-factory planning is also required.

Multi-factory production scheduling is a generalization of the single factory scheduling, which for the first time, is introduced by Williams [26]. This model is prevalent in today's supply chain environment. The distributed production approach enables to have flexibility because it makes factories closer to the customers,

Keywords. Scheduling, distributed system, multi-objective optimization, heuristic, elastic constraints method, Pareto front improving.

[1] Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran.
[2] Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, Tehran 15916-34311, Iran.
*Corresponding author: fatemi@aut.ac.ir

Article published by EDP Sciences

employ professionals, comply with local laws, reduce their costs, produce more effectively and respond to their market changes more quickly. However, each factory is considered as an individual entity that has different efficiency and is subject to particular constraints, *e.g.*, machine advances, labor skills and education levels, labor costs, government policies, taxation, nearby suppliers and transportation facilities. A practical example of multi-factory production problems can be found in the multinational lingerie company in Hong Kong [16], electric power generating industries [25], automotive manufacturers in Italy [12], food and chemical process industry [24] and steel corporation in the USA with four factories [21].

In this paper, a multi-factory production network problem with a set of factories and a set of demands for each of them is considered. Since the production processes within many industrial enterprises are distributed over several manufacturing factories, the factories themselves are responsible for the production of their regions. In such a system, a schedule should give enough flexibility to a local scheduler. This can be attained by transporting the jobs from the overloaded machine to the machine which has fewer workloads. Therefore, considering the transportation time from one factory to another is important in the scheduling process. This problem mainly concentrates on solving two issues simultaneously: (i) determining the most suitable factory for each job; and (ii) determining the scheduling of assigned jobs to each factory. For scheduling problems, in the first step, it is assumed each factory can produce the same quality of the product. Since customer demands correspond to the factories and each factory has its customers, the demands, however, can be satisfied by others by imposing some transportation times. This can be attained by transporting the jobs from among the production networks to guarantee a better objective function. In other words, when job transportation is allowed, a job can be executed in another factory. It is clear that considering this assumption in the scheduling makes sense and inevitably is more practical than those problems that do not take such an assumption into account. Here, it is also assumed that there are transportation routes among factories and a large number of transporters with the same quantity, capacity, time and cost. Absolutely, for such coordination, the communication between the entities of the network is essential.

Parallel factories and series factories are two types of structures in the distributed production network. In the parallel structure, the factories can be homogeneous or non-homogeneous. In the parallel structure, the allocation of demands to a suitable factory (especially in the heterogeneous environment) is a complex problem; therefore, the majority of literature is dedicated to homogeneous factories. Because the real-world distributed systems are usually heterogeneous, our research assumes that the distributed scheduling (DS) network consists of heterogeneous parallel factories in which each factory has identical machines and different factories have different speeds. Systems of such production networks have many real-life applications, *e.g.*, in semiconductor manufacturing, commonly, the newer and more modern machines have faster processing speeds compared to existing machines [9].

Interest in multi-objective scheduling has been increasing recently, but due to its complexity, the research in this area in comparison with the single criterion problems is limited. In a real production environment, several objectives frequently need to be taken into account, simultaneously. Among them, due-date related objective functions are one of the most critical factors [14]. In this class, the earliness and tardiness are significant in satisfying due dates [2] in which early completion may induce storage costs, while tardy completion may induce penalty costs. So, in this paper, multi-objective scheduling with a due-date related objective is considered. This paper aims to propose the mathematical-based exact algorithm to solve the problem. Using such an exact algorithm for the complex problems obtaining an optimal solution is difficult especially for the large-size instances. Since in reasonable computational time single factory scheduling with parallel machines and an earliness/tardiness objective function as a simple case of multi-factory scheduling is an NP-hard problem [10] and changing from a single objective to a multi-objective formulation is not a trivial task [13], our multi-objective problem with several factories is also NP-hard and consequently devising heuristics and metaheuristics, especially in the large-size instances, is highly desirable heuristic algorithms are more preferable and acceptable in practice, because they can obtain near-optimal solutions in a reasonable time. So this paper proposes a multi-objective heuristic algorithm and enriches it with a new methodology.

The paper continues with a comprehensive review of related research in Section 2. Section 3 presents a mathematical model. To solve the problem, Sections 4 and 5 propose a multi-objective algorithm and a heuristic approach, respectively. Section 6 is devoted to detailed computational results. Finally, Section 7 presents some conclusions and remarks about future researches.

2. Literature review

A limited amount of literature has been dedicated to this scheduling problem. Among them, it can be referred to Cicirello and Smith [7]. They applied wasp-like agents for distributed coordination with two factories and parallel machines. These agents used a model of wasp job allocation to determine the new job's acceptance of the machines' queue. This approach for jobs arriving is also used to determine their bidding strategies. They benchmarked the performance of their work on a real problem, which had to assign trucks to paint booths in a simulated vehicle paint shop. Carroll and Grosu [5] modeled selfish multi-user job scheduling as a non-cooperative and extensive-form game in the parallel identical machines environment. In this problem, the authors assumed that there are several users with multiple selfish jobs with the makespan of their own jobs. Due to the absence of coordination among the users, in that study, the price of anarchy to quantify the costs was computed. Terrazas-Moreno and Grossmann [23] dealt with simultaneous scheduling and planning problems in a production–distribution network of continuous multi-product factories. In this problem, the geographically distributed production network is made up of several production factories distributed in different markets. The authors proposed a hybrid bi-level and spatial Lagrangian decomposition method.

Recently, Behnamian and Fatemi Ghomi [3] considered the multiple factories scheduling problem in which all factories are located in a single site, and therefore, the jobs transportation times among them are neglected. In this problem, the factories available to process the jobs have different speeds in which each factory has identical parallel machines. To minimize the makespan of all jobs, they proposed mixed-integer linear programming (MILP) model, the longest processing time (LPT) based heuristic and genetic algorithm (GA). In this paper, after representing a matrix-based encoding scheme, they improved the proposed genetic algorithm with a local search algorithm. They also developed a lower bound and showed the proposed algorithms are efficient.

Shao *et al.* [22] addressed a distributed no-wait flow shop scheduling problem (DNWFSP) with the makespan criterion by using. They, firstly, investigated several speed-up methods based on the problem properties of DNWFSP to reduce the evaluation time of the neighborhood with O(1) complexity. Then they proposed an improved NEH heuristic to generate a promising initial solution. In this study, several neighborhood structures are employed to improve their proposed iterated greedy algorithms.

İnkaya and Akansel [15] considered coordinated scheduling of the transfer lots in an assembly-type supply chain that consists of at least two stages, where the upstream stages manufacture the components for several products to be assembled at the downstream stages. In order to enable faster flow of products through the supply chain and to decrease the work-in-process inventory, they used the concept of lot-streaming as a means of supply chain coordination. They also introduced a mathematical model and a genetic algorithm to minimize the sum of weighted flow and inventory costs. A backtracking search hyper-heuristic algorithm is proposed to solve distributed assembly permutation flow-shop scheduling problem by Lin *et al.* [17]. In the proposed algorithm, ten heuristic rules are designed to construct a set of low-level heuristics and the backtracking search algorithm is employed as the high-level strategy to manipulate the low-level heuristics. To generate a feasible schedule, they also proposed a solution encoding and decoding scheme.

Chang and Liu [6] proposed a hybrid genetic algorithm for solving the distributed and flexible job-shop scheduling problem with the makespan objective function. They offered a novel encoding mechanism to solve invalid job assignments and employed a genetic algorithm to solve the flexible job-shop scheduling problems. Also, in the proposed hybrid genetic algorithm, various crossover and mutation operators are used to increase the probability of finding the optimal solution and diversity of chromosomes.

Wu *et al.* [27] studied the distributed assembly flexible job shop scheduling problem. This problem can be decomposed into several flexible job shop scheduling problems and several single machine factory scheduling

problems. To minimize the earliness/tardiness and the total cost simultaneously, they proposed a mixed-integer linear programming. An improved differential evolution simulated annealing algorithm was also introduced in which to select the offspring, the authors applied a greedy idea combined with the non-dominated sorted selection. Zhang and Xing [28] addressed the distributed flowshop scheduling problem with finite buffer and makespan criterion. The considered problem consists of multiple homogeneous factories and each one is set as a permutation flowshop with limited buffers between any two adjacent machines. They proposed two constructive heuristics for generating a good initial solution for metaheuristics and used a differential evolution algorithm to improve it. Meng *et al.* [18] introduced the customer order constraint into the distributed permutation flowshop scheduling problem. In this problem, it is assumed a set of customer orders needs to be manufactured in a number of factories and each order composed of some defined jobs should be processed in the same factory. To minimize makespan among factories, they proposed a mathematical model. Then, they developed three metaheuristics, namely, a variable neighborhood descent, an artificial bee colony and an iterated greedy.

This review reveals that:

(i) A large percentage of the reviewed papers considered job shop scheduling and only a small fraction of the literature tackles parallel machines scheduling.
(ii) A large part of the literature is devoted to homogeneous factories in the distributed scheduling problem.
(iii) The majority of researchers in multi-factory scheduling are interested in proposing heuristics and metaheuristics in which a large number of them are GA-based metaheuristics.
(iv) Most studies are devoted to single-objective optimization.

Note that, the real-world distributed systems usually are heterogeneous. In other words, each factory can be considered as an entity that has various efficiency and is subject to different constraints, *e.g.*, machine advances, labor skills/costs and education levels, government policy, tax, nearby suppliers, and transportation facilities. According to the above points, it can be concluded that distributed scheduling with parallel machines and heterogeneous factories is an interesting subject. Also, in a real production environment, several objectives frequently need to be considered simultaneously. Therefore, the multiple objectives in multi-factory scheduling are also embedded in the present paper.

3. Problem description and mathematical model

In this paper, a multi-factory production network problem with a set of parallel factories and a set of demands for each of them is considered. Since customer demands correspond to the factories and each factory has its own customers, the demands, however, can be satisfied by others by imposing some transportation times. This can be attained by transporting the jobs from the overloaded machine in the origin factory to the machine which has fewer workloads in the substitute factory to guarantee a better objective function. In this research, it is assumed that the distributed scheduling network consists of heterogeneous parallel factories in which each factory has identical machines and different factories have different speeds. In this paper, multi-objective scheduling with a due-date related objective was considered, in which total completion times, as well as the sum of the earliness and tardiness of jobs must be minimized, simultaneously.

The number of independent factories forming the production network is denoted by F which has a specific local job cluster. Each factory f owns a cluster Cl^f and has m^f identical parallel machines with speed v^f. Job i that belongs to factory cluster f performs on factory q and is denoted by j_i^{fq}. If $f = q$, the job is performed locally, otherwise it is migrated and the transportation time between factory f and q must be considered. The job set performed by factory f is denoted by n^f. If factory q performs all its local jobs, then $Cl^f = n^f$. With such definitions, $n = \sum_{f=1}^{F} n^f = \sum_{f=1}^{F} Cl^f$ is the number of jobs that must be scheduled on F factories.

Now the proposed model is constructed according to the following assumptions:

(1) All data used in the paper are known deterministically when scheduling is undertaken.
(2) There are n independent jobs that are available at time 0.
(3) Machines are available at all times if they are not busy, with no breakdown.

(4) A job, once started on the machine, must be completed on it without interruption.
(5) Orders quantity will not change after getting released from customers in each region.
(6) Two objective functions are considered in the problem definition, *i.e.*, the sum of the earliness and tardiness (ET) of jobs and total completion time (TCT).
(7) The factories have a parallel structure and are heterogeneous, *i.e.*, the production processes performed in different factories are similar and each factory has identical machines, but the machines in the different factories may have different speeds.
(8) Jobs can be transported between factories and no loss and damage occur during transportation between the factories and always, an infinite number of identical transporters are ready.
(9) Loading/unloading times are not considered separately and included in transportation times.

The input parameters and decision variables are defined below.

n number of true jobs to be scheduled
i, j, k index of jobs; $i, j, k \in \{1, 2, \ldots, n\}$
f, q index of factories; $f, q \in \{1, 2, \ldots, F\}$
m^f number of parallel machines in factory f
n^f number of assigned jobs to factory f
p_i processing time of job i
E_i earliness of job i
T_i tardiness of job i
d_i due date of job i
v^f the relative speed of machines in factory f
t^{fq} the transportation time to carry the jobs from factory f to factory q
p_i^{fq} modified processing time of job i in factory q which originally is ordered to factory f
C_i^f completion time of job i in factory f
L large positive number

$$x_{ij}^f = \begin{cases} 1 \text{ if job } i \text{ is scheduled immediately before job } j \text{ in factory } f, \\ 0 \text{ otherwise.} \end{cases}$$

$$y_i^f = \begin{cases} 1 \text{ if job } i \text{ is assigned to factory } f, \\ 0 \text{ otherwise.} \end{cases}$$

$$w_i^f = \begin{cases} 1 \text{ if job } i \text{ is orginally in cluster of factory } f, \\ 0 \text{ otherwise.} \end{cases}$$

It is important to notice the dummy jobs 0 and $n+1$ are introduced and that their processing times are 0. The following model translates the problem assumptions into the mathematical formulation using the above nomenclature.

$$Z = \text{Min} \left(\sum_{f=1}^{F} \sum_{i=1}^{n} C_i^f, \quad \sum_{i=1}^{n} E_i + T_i \right) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{f=1}^{F} y_i^f = 1, \qquad i = 1, 2, \ldots, n, \tag{3.2}$$

$$\sum_{\substack{i=0, \\ i \neq j}}^{n} \sum_{f=1}^{F} x_{ij}^f = 1, \qquad j = 1, 2, \ldots, n, \tag{3.3}$$

$$\sum_{j=1}^{n} x_{0j}^f = m^f, \qquad f = 1, 2, \ldots, F, \tag{3.4}$$

$$\sum_{f=1}^{F} x_{0j}^{f} \le 1, \qquad j = 1, 2, \ldots, n, \tag{3.5}$$

$$\sum_{\substack{i=0,\\ i\neq j}}^{n} x_{ij}^{f} + \sum_{\substack{k=1,\\ k\neq j}}^{n+1} x_{jk}^{f} = 2 \cdot y_{j}^{f}, \qquad j = 1, 2, \ldots, n, \qquad f = 1, 2, \ldots, F, \tag{3.6}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} \sum_{f=1}^{F} x_{ij}^{f} \le 1, \qquad i = 1, 2, \ldots, n, \tag{3.7}$$

$$\sum_{f=1}^{F} \left(x_{ij}^{f} + x_{ji}^{f} \right) \le 1, \qquad i = 1, 2, \ldots, n-1, \qquad j > i, \tag{3.8}$$

$$p_{i}^{fq} = w_{i}^{f} \left(\left(\frac{p_i}{v^q} \right) + 2t^{fq} \right), \qquad i = 1, 2, \ldots, n, \qquad f, q = 1, 2, \ldots, F, \tag{3.9}$$

$$C_{j}^{f} - C_{i}^{f} \ge p_{j}^{fq} - L \left(1 - x_{ij}^{f} \right), \qquad i, j = 0, 1, \ldots, n, \qquad i \neq j, \quad f = 1, 2, \ldots, F, \tag{3.10}$$

$$C_{i}^{f} + E_i - T_i = d_i, \qquad i = 1, 2, \ldots, n, \qquad f = 1, 2, \ldots, F, \tag{3.11}$$

$$E_i \ge d_i - C_{i}^{f}, \qquad i = 1, 2, \ldots, n, \qquad f = 1, 2, \ldots, F, \tag{3.12}$$

$$T_i \ge C_{i}^{f} - d_i, \qquad i = 1, 2, \ldots, n, \qquad f = 1, 2, \ldots, F, \tag{3.13}$$

$$x_{ij}^{f}, y_{i}^{f} \in \{0, 1\}, \qquad i, j = 0, 1, \ldots, n+1, \qquad i \neq j, \quad f = 1, 2, \ldots, F, \tag{3.14}$$

$$C_{i}^{f} \ge 0, \qquad i = 1, 2, \ldots, n, \qquad f = 1, 2, \ldots, F, \tag{3.15}$$

$$E_i \ge 0, \qquad i = 1, 2, \ldots, n, \tag{3.16}$$

$$T_i \ge 0, \qquad i = 1, 2, \ldots, n. \tag{3.17}$$

In the proposed mathematical model, constraint (3.1) is the objective function. Constraint (3.2) indicates that job i requires only one factory for its processing. Constraint (3.3) implies that a job j is either the first job (when $i = 0$) or has exactly one predecessor (job i) on one of the F factories. With assumption $n \ge m^f$, constraint (3.4) ensures that there are as many dummy jobs 0 on each of the m^f machines of every factory f. Constraint (3.5) guarantees that job j cannot be the first job on a machine of more than one factory. Constraint (3.6) controls that every job can be either a successor or predecessor on each machine in the factory to which it is assigned. Constraint (3.7) shows that every job has at the most one succeeding job. Constraint set (3.8) states that a job cannot be at the same time both a predecessor and a successor of another job. Relation (3.9) modifies the processing time of job i according to the distance between the original factory of the job which belongs to it and the factory that the job is finally processed on and the speed of machines in the destination factory. Constraint (3.10) establishes the relationship between the completion times of jobs i and j assigned to the same machine. Constraint (3.11) indicates the relation among the completion time, due date, earliness and tardiness of a job. Constraint (3.12) calculates the earliness of the jobs. Constraint (3.13) calculates the tardiness of the jobs. Constraints (3.14) to (3.17) represent the state of the variables and parameters used in our model. Note that the value of C_{i}^{f} is zero when the job i is not assigned to factory f.

4. Multi-objective optimization

Since our model is a multi-objective problem (MOP), this section is devoted to MOP concepts. The most important methods in the multi-objective optimization are weighted sum and ϵ-constraint methods. These methods are very efficient and simple approaches that have disadvantages despite their wide usages. The most important positive property of the weighted sum method is that it requires the same computational effort as

the single objective version of the MOP to solve. However, this method has some drawbacks, *e.g.*, with a large number of objectives, it is very time-consuming and, in some cases, dependent on the form of Pareto front, it cannot generate all efficient solutions [8]. Furthermore, this method is unable to deal with the non-convex Pareto fronts. In the second method of the MOP, the ϵ-constraint method, all efficient solutions can be found by appropriately specifying the upper bounds. However, determining the upper bound is usually NP-hard. The scalarized problem in the ϵ-constraint method is harder than the single objective version of the MOP. Furthermore, this method will strongly require *a priori* knowledge of the problem, and it is not appropriate for a large number of objective functions [1].

Now, it can be concluded that almost all the methods mentioned above have some drawbacks. In the following subsection, the new method is introduced.

4.1. Elastic constraints method

In the elastic constraints method, the advantages of all the above methods are combined to avoid their drawbacks. The scalarization of the problem is as follows in which the original feasible region of scheduling problem is S and objective functions are f_a and f_b.

$$\begin{aligned} \min \quad & w_a f_a(x) + w_b s_b \\ \text{s.t.} \quad & x \in S \\ & f_b(x) + l_b - s_b = \epsilon_b \\ & w_a + w_b = 1 \\ & 0 < w_a,\, w_b < 1 \\ & l_b \geq 0,\, s_b \geq 0 \end{aligned} \tag{4.1}$$

where w_a is the importance of the objective function a, ws are the penalty coefficients, ϵ_b is a right-hand side of the objective function constraint and s_b and l_b are surplus and slack variables, respectively. In this method, the appropriate selection of s_b and l_b in the elastic constraint method causes to turn the f_b upper bound into an equality constraint and penalizes the constraint violation by w_b.

Some important properties of this method are summarized as follows.

Theorem 4.1. *Let $w > 0$ and (x^*, l^*, s^*) be an optimal solution of (4.1). Then x^* is a weakly efficient solution for the MOP.*

Proof. If we assume $w > 0$, then we will have $s_k^* = \max\{0, f_b(x^*) - \epsilon_b\}$. Assume there is some $x' \in S$ such that $Z(x') < Z(x^*)$. If assume $l_b' := \max\{0, \epsilon_b - f_b(x')\}$ and $s_b' := \max\{0, f_b(x') - \epsilon_b\}$, then (x', l', s') is feasible for (4.1) and has a better objective function value than (x^*, l^*, s^*) then it contradicts the optimality of (x^*, l^*, s^*). □

Theorem 4.2. *Let $w > 0$ and (x^*, l^*, s^*) be an optimal solution of (4.1). Then uniqueness of x^* in all optimal solutions and $s^* > 0$ are two sufficient conditions for x^* such that it is efficient.*

Proof. If assume $s^* > 0$ and there is $x' \in S$ such that $Z(x') < Z(x^*)$, then Theorem 4.1 gave a reason that x^* is weakly efficient and it is obvious that the first condition holds. Also, simultaneous holding of $s^* > 0$ and $w > 0$ imply that $l^* = 0$. According to Theorem 4.1 and with defining s' and l', we will have a feasible solution (x', l', s') such that $w_a f_a(x') < w_a f_a(x^*)$ or $s_b' < s_b^*$ then it contradicts the optimality of $(x^*, 0, s^*)$. □

Theorem 4.3. *Since the elastic constraints method comprises both the weighted sum and ϵ-constraint method,* (i) *if $\epsilon_b \leq \min\{f_b(x) : x \in S\}$, problem (4.1) is equivalent to a weighted sum problem; and* (ii) *if $w_b = \infty$, problem (4.1) is equivalent to a ϵ-constraint problem.*

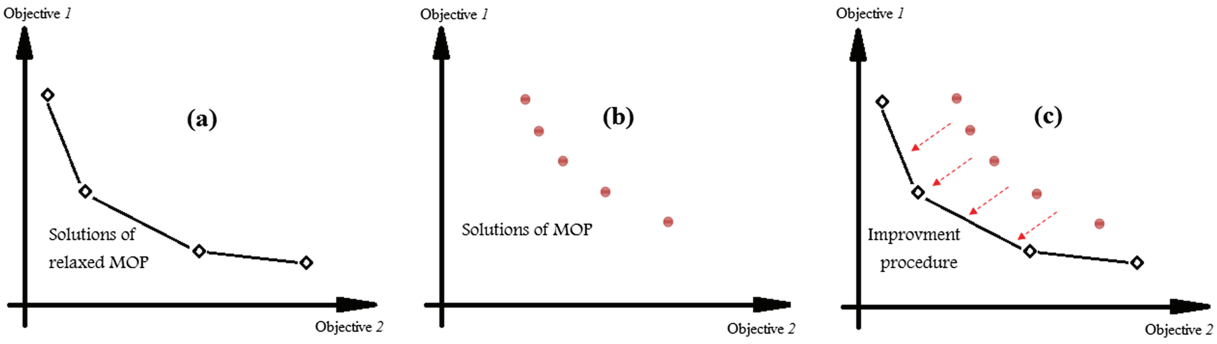

FIGURE 1. Solution guide mechanism.

Proof. (i) By choice of $\epsilon_b \leq \min\{f_b(x) : x \in S\}$, any feasible solution of (4.1) must have $s_b \geq 0$. In the optimal solution, additionally, l_b is 0. Thus, $s_b = f_b(x) - \epsilon_b \geq 0$ and solving (4.1) is equivalent to solving "$\min w_a f_a(x) + w_b f_a(x) - w_b \epsilon_b$". Since the last term is constant, this is a weighted sum problem.

(ii) By choice of $w_b = \infty$, since s_b must be zero in the objective function minimization, any solution (x, l, s) of (4.1) with finite objective function value must satisfy $f_b(x) + l_b = \epsilon_b$, *i.e.*, $f_b(x) \leq \epsilon_b$ with the objective function "$\min w_a f_a(x)$". Since the coefficient w_a has no effect on the model solutions, problem (4.1) is equivalent to the ϵ-constraint problem.

□

4.2. Improvement of Pareto front

In this subsection, to improve the obtained Pareto front of the elastic constraints method will be introduced. In the literature, such a method is named the solution guiding method. In our proposed method, the new Pareto optimal solutions (PS) are generated according to the information already obtained from solutions and this procedure is repeated until no new Pareto optimal solutions are found. Figure 1 shows the details of this procedure.

The proposed algorithm has the following steps in details:

Step 1. Relax the binary variables and solve the MOP using an elastic method. Find the set of best achievable efficient solutions (BAS).

Step 2. Create and solve the MOP with the elastic multi-objective method.

Step 3. Confine the search's zones and solve the specific sub-problem iteratively in the confined space (CS) to find the new PS, as shown in Figure 2b.

To create the confined space, it is assumed that N Pareto optimal solutions have been found in Step 2. For each couple-solution on the front, if the PS are arranged in decreasing order of z_a, the following model denoted by CP(α) for αth and $(\alpha+1)$th PS with the confined objective values were solved:

$$\begin{aligned} \min \quad & \lambda_a f_a(x) + \lambda_b s_b \\ \text{s.t.} \quad & x \in S \\ & f_a(x) \leq z_a(\alpha) + \epsilon \\ & f_b(x) + l_b - s_b = z_b(\alpha+1) \\ & \lambda_a + \lambda_b = 1,\ 0 < \lambda_a,\ \lambda_b < 1,\ l_b \geq 0,\ s_b > 0,\ \epsilon > 0. \end{aligned} \tag{4.2}$$

Considering two cases $\alpha = 0$ and $\alpha = N$, totally $N+1$ problems must be solved. The constant term "ϵ" is required to prevent the generation of solutions previously found. If problem CP(α) proved to be infeasible, it means that no new PS can be found in the zone determined by solutions α and $\alpha+1$ (the gray area in

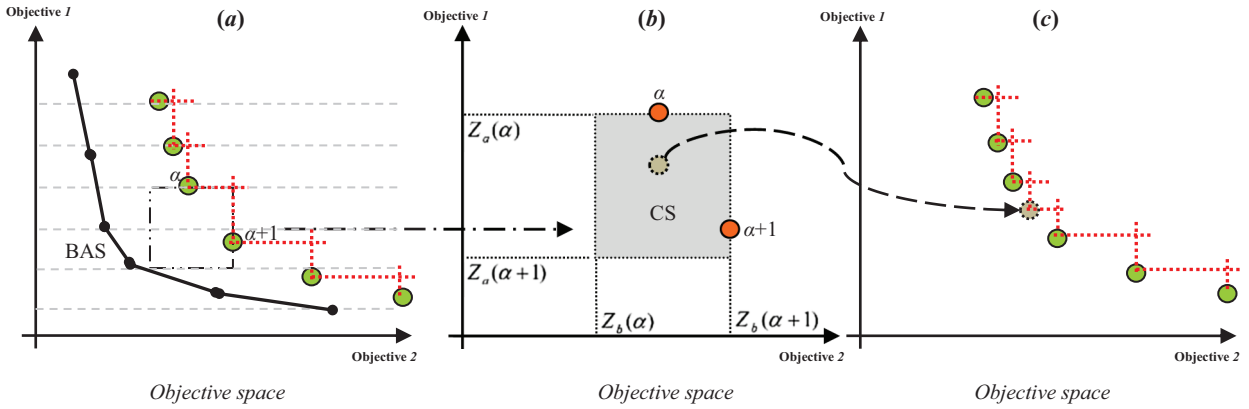

FIGURE 2. Example of new PS discovered with the solution guide mechanism.

Fig. 2b). In the case that the new PS is found, go to Step 4a, else, go to Step 4b. Note that, since the best achievable values limit the values of $z_a(\alpha+1)$ and $z_b(\alpha)$, it is not required to confine their values in CP(α).

Step 4a. According to the iterative procedure, if the new PS can be found, insert them in the last Pareto set for the next round of search. By "merging" and update the Pareto front and go to Step 3. Now, at least a new couple will be obtained and it is also possible that the new PS dominates some old solutions and the newer couples will be achieved.

Step 4b. If, in the last round of the algorithm, the specific zone determined by two adjacent solutions is explored without giving the new PS, it is necessary to repeat exploration in the next round. Since these zones gave no new PS in the previous runs, they must be removed from the list of PS that will be explored in the next round. Therefore, put the current couple in the tabu list until all couples of this round are checked and repeat Step 3.

As mentioned earlier, all single factory scheduling problems in the multi-factory environment are NP-hard and no exact algorithm can be designed, especially for large-size instances. Solving the scheduling problem in the real and large-sizes instances, where it is more complicated to be solved exactly, induces us to devise a heuristic algorithm. Thus, in the next section, a new heuristic algorithm for the problem will be proposed.

5. HEURISTIC ALGORITHM: MULTI-STRUCTURE LOCAL SEARCH ALGORITHM

Population-based heuristics handle a "population" of solutions rather than a single feasible solution. In general, these algorithms start with an initial population and use some principles (such as a mutation in GA), and cooperation in exchange information between individuals (such as "pheromone" in an ant colony) to improve the initial population quality. Since members of the population contribute to the evolutionary process and the generation mechanism is parallel, the population-based methods are attractive for solving the problems. In contrast, in neighborhood search algorithms, the generation of solutions relies upon one individual solution and its neighbors. By using the neighborhood structure mechanism, the solving procedure iteratively projects the neighbors into the objective space in a specific search direction by optimizing the corresponding objective function. Basically, a local search algorithm carries out the exploration within a limited region of the search space and facilitates the finding of a better solution without doing further investigation. This procedure is repeated to diversify the search directions [11].

Surveying a variety of heuristics and metaheuristics used to solve combinatorial problems gives us a good idea to propose a new algorithm. In this section, a new high-level local search algorithm which to select a candidate solution makes use of six choices in three levels is introduced. This algorithm is called a *multi-structure local*

search (MSLS) algorithm, and since essentially it is designed based on the structure of our problem, it is expected that this heuristic to have good performance when solving the large-size instances. The MSLS is a fast and effective search procedure that produces a systematic change in the neighborhoods.

As shown in Figure 3, an ordinary MSLS algorithm starts with an initial solution, $x \in S$, where S is the entire search space, and manipulates it through three-nested loops in which the search alters and explores *via* three main structures, so-called "one-machine local search (OLS)", "one-factory local search (OFS)" and "whole-local search (WLS)". The loops of WLS and OFS work as a refresher reiterating the OLS loop, while the OLS loop carries out the major local search. In other words, OLS explores an improved solution within the local neighborhood, while WLS and OFS diversify the solution by switching to another local neighborhood. Once an inner loop is completed, the outer loop reiterates until the termination condition is met.

In the following subsection, several neighborhood structures with notation L_s are defined.

5.1. Local searches

Here, three local search structures *i.e.*, OLS with notation $L_1^l (l = 1, 2, 3)$ and L_2, OFS with notation L_3 and WLS with notation L_4 that are created in the following manner and used in our proposed algorithm:

(1) For L_1^1:
- Choose machine i randomly from the randomly selected factory f,
- Choose two jobs j_1 and j_2 on machine i, randomly,
- Swap jobs j_1 and j_2.

(2) For L_1^2:
- Choose machine i randomly, from the randomly selected factory f,
- Choose job j and a valid position "*pos*" on machine i, randomly,
- Transfer job j to position *pos*.

(3) For L_1^3:
- Choose machine i randomly, from the randomly selected factory f,
- Choose cutting point randomly on machine i to divide the sequence of job into two parts,
- Swap two parts.

(4) For L_2:
- Choose two machines i_1 and i_2 randomly from the randomly selected factory f,
- Choose job j_1 in i_1 and job j_2 in i_2 randomly,
- Swap jobs j_1 and j_2.

(5) For L_3:
- Choose job j_1 and machine i_2 randomly, where j_1 does not belong to i_2 from the randomly selected factory f,
- Choose a valid position "*pos*" randomly in i_2 in the factory f,
- Transfer job j_1 to i_2 at position *pos*.

(6) For L_4:
- Choose job j_1 randomly from the randomly selected factory f and machine i_2 from the randomly selected factory q, where j_1 does not belong to i_2,
- Choose valid position "*pos*" randomly in i_2 in the factory q,
- Transfer job j_1 to i_2 at position *pos*.

Note that in the MSLS, the algorithm always tries to use the fastest local search that available first. If, after an iteration, a new solution is found, then another neighborhood is used (in L_s the value of s is incremented), and every time no improvement is made, the first and the fastest local search is used ($s = 1$).

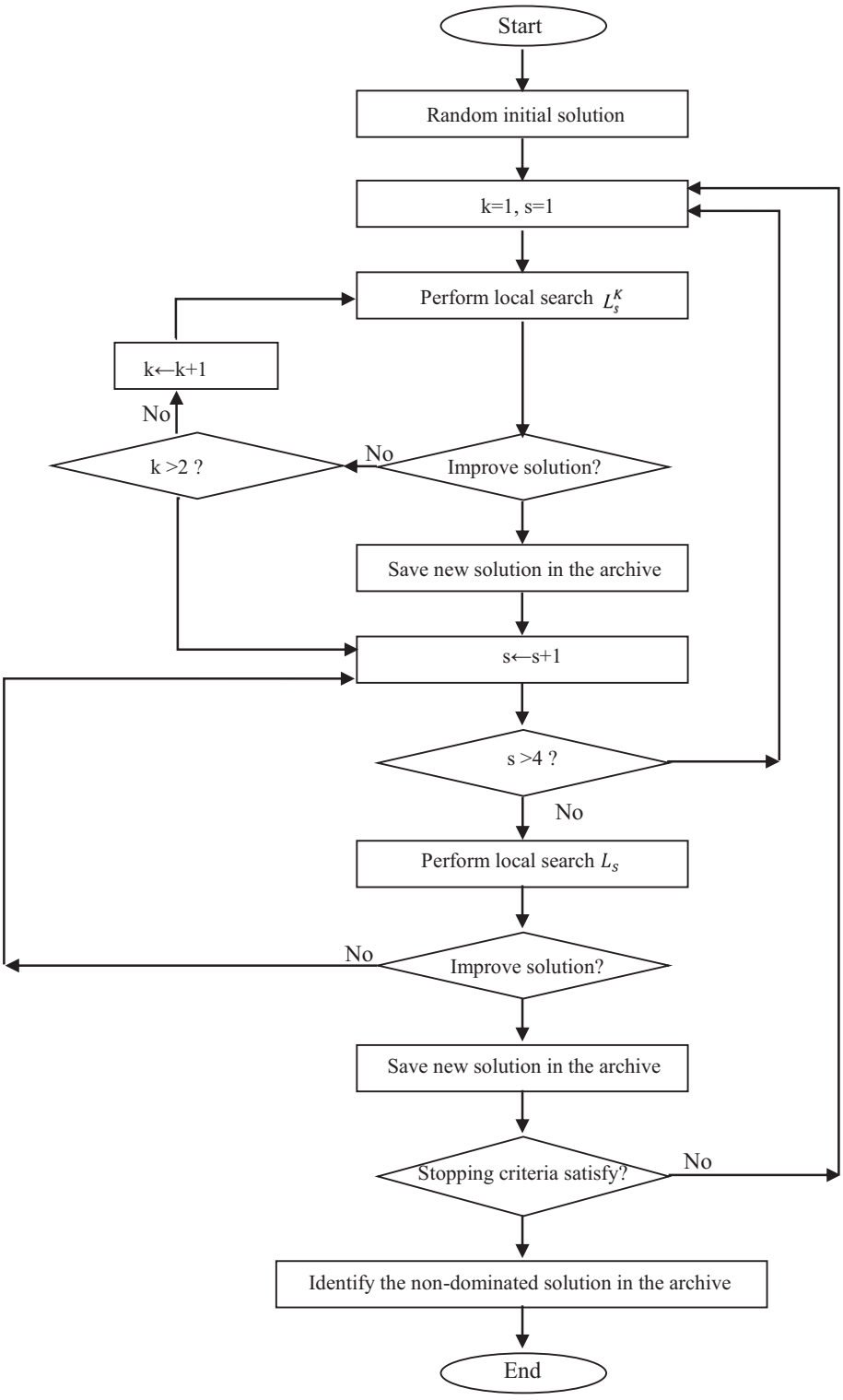

FIGURE 3. The flowchart of multi-structure local search algorithm.

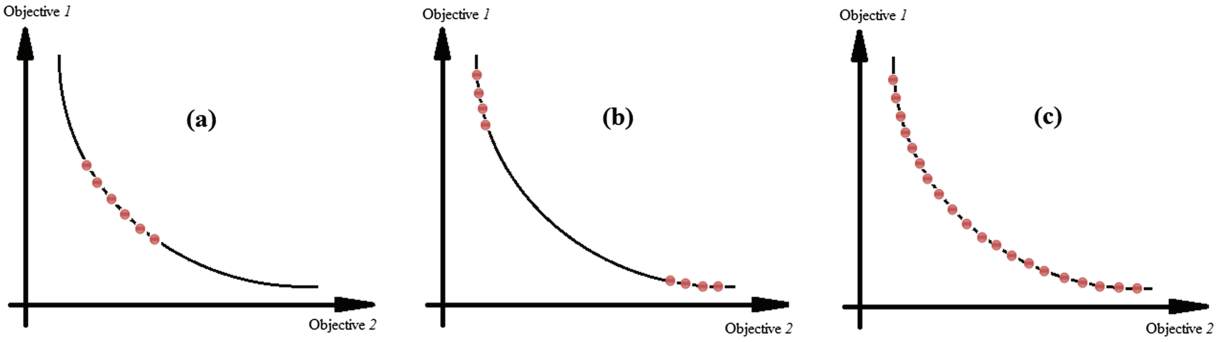

FIGURE 4. The spread of Pareto solutions on Pareto front.

5.2. Improvement of Pareto front in MSLS

After running the MSLS algorithm, one of the solution's quality index, the spread of solutions, may not be appropriate. The aim of this subsection is to propose a moving guide algorithm that improves the results of the MSLS.

5.2.1. Diversity of solutions

The diversity and spread of solutions are two critical issues in multi-objective optimization. As shown in Figure 4, typically, the spread of Pareto solutions on the Pareto front has three cases, including (a) the solutions of the Pareto front have inadequate diversity; (b) the solutions of the Pareto front are not explored and finally; and (c) the ideal solutions.

5.2.2. Goal-oriented guide

The main objective behind choosing a non-dominated guide for a dominated solution is to search the neighborhood of a dominated solution so as to improve its position in order to relocate in the Pareto front. By following the nearest non-dominated guide, a dominated solution is likely to obtain a better solution. This helps to gain a better Pareto front. In the guidance of a dominated solution, as shown in Figure 5, there are several methods for targeting, included (a) a dominated solution can follow a dominated or non-dominated solution; and (b) a dominated solution follows the nearest non-dominated solution. A weakness of this method is that when one solution dominates all other solutions, targeting toward a single target causes the diversity of search to be lost. To eliminate the drawbacks of these methods, a case (c) was proposed in which a dominated solution can follow either non-dominated or dominated solutions in its own region.

Now, the purpose of this subsection is to provide a technique with a low computational cost for the situation that the results are far from the ideal solutions. The goal of this technique is to improve the dominated solution in order to locate on the Pareto front by confining the solution space. The proposed technique uses a new approach for generating the set of potential guides from the available dominated solutions in each iteration. Multi-objective optimization requires a search for multiple targets to locate solutions closer to the Pareto optimal front and improve diversity. As shown in Figure 6, in the proposed technique, the solution space is confined in the zone CS, and this creates an opportunity to maintain the search procedure in those regions that are closer to the Pareto-approximation front.

Typically the dominated solutions are not considered in the obtained results, but this important issue should be taken into account since some of these solutions have special properties such as being located in the sparse part of the Pareto front. In this part of the algorithm, if it is necessary, some of these solutions will be used. Before proposing a technique, in detail, analyzing the distribution of the obtained Pareto solution by MSLS and determining in what zones the spread of solutions is not appropriate is needed. Figure 6 illustrates the process

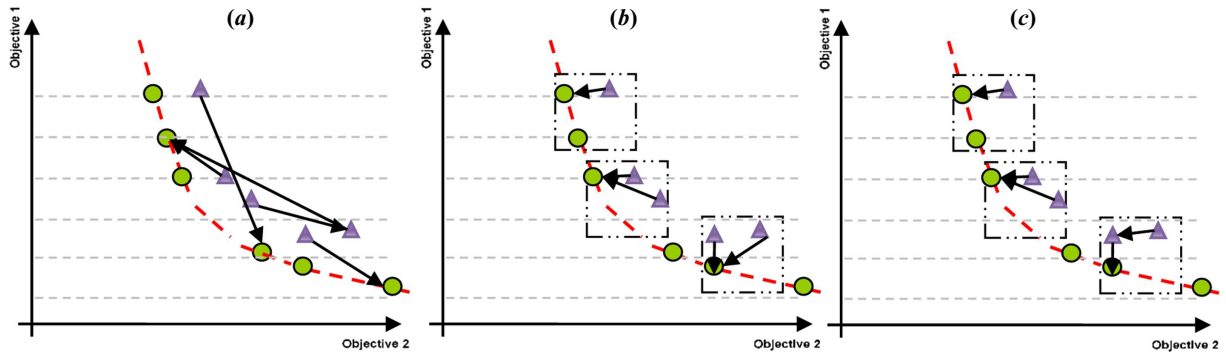

FIGURE 5. Some typical search features of the solution guide.

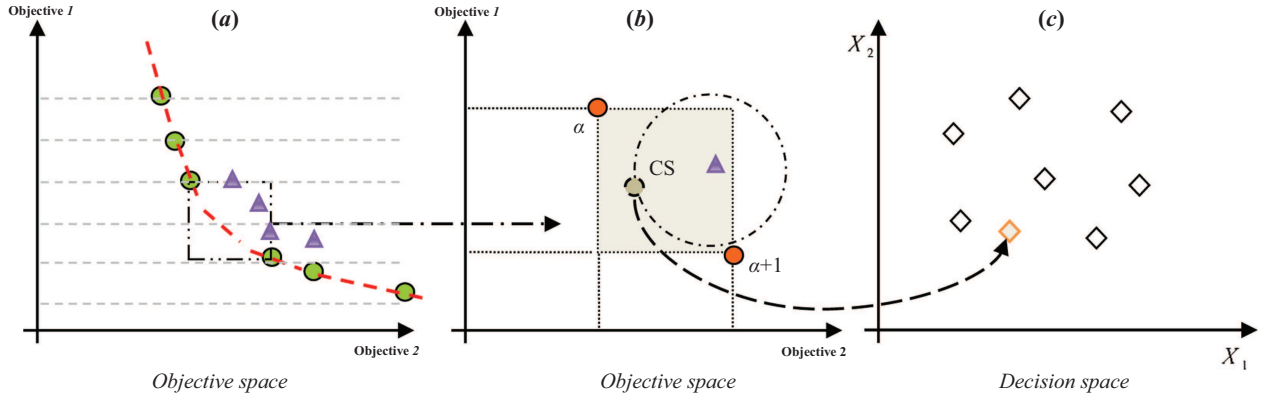

FIGURE 6. Example of new PS discovered with the goal-oriented guide.

of the *goal-oriented guide technique* in which the circles and triangles represent the non-dominated and dominated solutions, respectively. It is seen that the non-dominated solutions are missing in some zones.

In this regard, a search direction is chosen randomly for each dominated solution among the solutions located in its zone. The PS follow their respective non-dominated or dominated solution to maintain the search space within their zones. Three particular cases may occur: (i) by excluding the selected non-dominated solution, if the corresponding zone is completely empty, the nearest dominated or non-dominated solution from the nearest nonempty neighbor is selected; (ii) if the corresponding zone has a one dominated solution, this solution is guided toward the node by extrapolating (or interpolating) from nearest existing solutions in the nearest neighbor of the sparse zone; and (iii) if the corresponding zone is completely empty, either a non-dominated or dominated solution from other zones is chosen and it is guided as described in (ii). This procedure is continued until the Pareto front is close to the ideal shape or stopping criteria is met. Note that, for moving guide in this step, the crossover-like operators can be used.

6. Computational results

The literature review indicates that there is no study on the bi-objective multi-factory scheduling problem with heterogeneous factories. Therefore, the present paper seems to be the first study in this field. So, the improved MSLS is compared with the adaptations of an improved genetic algorithm recently proposed by Behnamian and Fatemi Ghomi [3] for a set of randomly generated test problems. In that study Behnamian

TABLE 1. GA parameters tuning [3].

Parameters	Problems		
	Small	Medium	Large
C_r	0.70	0.60	0.70
M_r	0.10	0.10	0.04
P_{size}	200	200	200

and Fatemi Ghomi [3] have applied the parameters tuning for the crossover rate (C_r) mutation rate (M_r) and population size (P_{size}). Table 1 shows the results for different sizes of problems; small medium and large.

In this paper, the proposed algorithms are run ten times independently. The CPLEX 7.0 software is also used to solve the mathematical model. The improved MSLS and GA were implemented in Borland C++ 5.02 and run on an Intel Pentium IV dual-core 2.00 GHz PC with 1022 MB RAM running Microsoft Windows 7.

In multi-objective optimization, it is important to decide how the quality of solutions is evaluated because the conflicting and incomparable nature of some of the criteria makes this process more complex. So, in this section, firstly, the evaluation metrics are introduced and then the numerical results are reported in two separate subsections: MILP on the small-size instances and MSLS on large-size instances.

6.1. Evaluation metrics

For evaluating the obtained solutions, in this paper, three indices as follows [4] were used. Note that, lower values of MID and RAS are preferred, but a higher value of SNS is better (more diversity in the obtained solutions is preferred).

(1) MID (mean ideal distance): The closeness between the Pareto solution and ideal point (0, 0) which is defined as follows.

$$\text{MID} = \frac{\sum_{i=1}^{n} c_i}{n} \tag{6.1}$$

where n is the number of non-dominated solutions and $c_i = \sqrt{f_{1i}^2 + f_{2i}^2}$.

(2) SNS: The spread of non-dominated solutions, as a diversity measure, can be expressed by the following relation:

$$\text{SNS} = \sqrt{\frac{\sum_{i=1}^{n} (\text{MID} - c_i)^2}{n-1}}. \tag{6.2}$$

(3) RAS: The rate of achievement to two objectives simultaneously which is represented in relation (23).

$$\text{RAS} = \frac{\sum_{i=1}^{n} \left(\frac{f_{1i}-F_i}{F_i}\right) + \left(\frac{f_{2i}-F_i}{F_i}\right)}{n} \tag{6.3}$$

where $F_i = \min\{f_{1i}, f_{2i}\}$.

6.2. Numerical results and discussion

In this subsection, the developed MILP, MSLS and GA are evaluated on two sets of instances. The first set is the small-size instances and designed to examine the effectiveness of the mathematical model. In the second one, the performance of the MSLS against the GA on large-size instances is examined.

Here, the important issue is the due dates of the jobs. To generate the due date of job that belongs to cluster, the following formula is proposed that it generates very tight due dates in which *random* is a random number from a Uniform distribution over the range (0, 1) and cm^f is $\max\{1, Cl^f/m^f\}$.

$$d_i = (0.1 + \text{random}) \times \left(cm^f/v^f\right) \times (p_i), \quad i = 1, 2, \ldots, n, \quad f = 1, 2, \ldots, F. \tag{6.4}$$

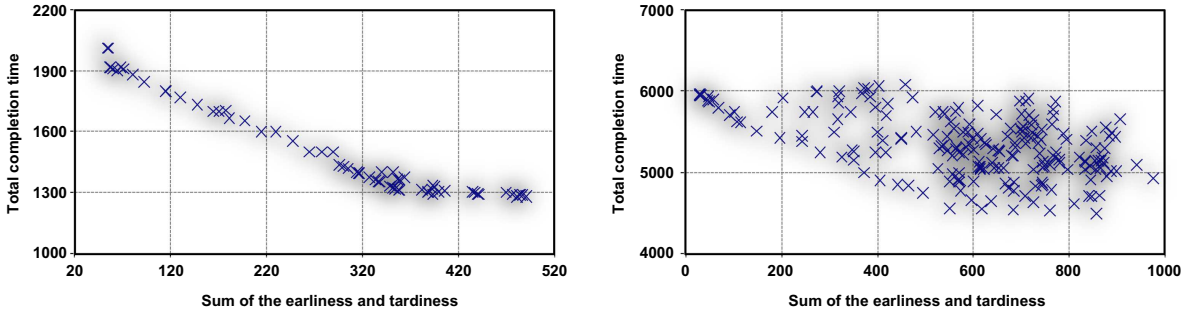

FIGURE 7. Pareto solutions yielded by MILP and elastic method implemented by CPLEX. (a) Instance with 10 jobs on 2-factory. (b) Instance with 15 jobs on 3-factory.

TABLE 2. Factor levels.

Factor	Levels			
Number of jobs	50	100	200	500
Number of factories	2	3	5	
Number of machines in each factory	2	5		
Standard processing times	U (50, 70)	U (70, 100)		

6.2.1. MILP on the small-size instances

Computational experiments are carried out to compare the results of MILP on the small-size instances and demonstrate that the proposed model can find Pareto optimal solutions. Here, the MILP model using the CPLEX solver is tested in two instances. The processing times of each instance are generated using a discrete Uniform distribution from 30 to 60.

Figure 7 provides the results of the 10-job with 2-factory and 15-job with 3-factory problems, respectively. From the computational results shown in Figure 7, it can be observed that the model can find near-optimal solutions efficiently. This method is, however, a computationally intensive procedure and is feasible only for the small-size instances.

6.2.2. MSLS on large-size instances

Following Behnamian and Fatemi Ghomi [3], data required for a problem consists of the number of jobs, the number of factories, and the number of machines in each factory and the range of processing times that their levels are listed in Table 2.

In general, all combinations of these levels are tested. However, some further restrictions are introduced. Due to its complexity, the instance with five factories and five machines in each factory is ignored. So, there are 40 test scenarios and ten data sets are generated for each one.

For a fair comparison between algorithms, similar to Ruiz and Stuetzle [20] and Ramanauskas *et al.* [19] studies, we have allocated equal time for the algorithms. Furthermore, due to several runs, we experimentally are aware that computational time has a direct relation with the number of factories, the number of jobs and the number of machines in each factory. So based on researches in the literature, the stopping criterion used for all algorithms is set to a computation time (CPU time) limit fixed to $\left(n \times \left(F \sum_{f=1}^{F} m^f\right)^{1/2}\right)$ seconds for algorithms. This stopping criterion is not only responsive to the number of factories but also is sensitive towards the rise in the number of jobs and the number of machines at each factory.

TABLE 3. Evaluation of obtained solutions for large size instances.

Jobs	Instance	Algorithm and index					
		MID		SNS		RAS	
		GA	MSLS	GA	MSLS	GA	MSLS
50 jobs	1	21 118.75	13 051.95	989.371	87.513	0.167	0.508
	2	24 022.86	12 920.87	1652.153	66.530	0.266	0.721
	3	25 668.02	12 781.08	1275.201	98.039	0.153	0.753
	4	23 058.26	12 728.94	1367.013	91.000	0.184	0.724
	5	15 912.18	13 047.19	697.816	76.567	0.614	0.824
	6	20 045.61	13 287.49	837.968	86.899	0.120	0.386
	7	21 628.71	13 440.03	1052.473	87.775	0.116	0.493
	8	22 372.91	13 087.83	1162.374	80.854	0.158	0.574
	9	23 079.07	13 298.39	1264.984	95.378	0.157	0.491
	10	24 147.23	13 107.09	1371.245	85.819	0.172	0.677
100 jobs	1	59 033.83	50 523.48	447.815	176.549	0.082	0.517
	2	59 569.68	50 152.49	325.814	318.709	0.109	0.378
	3	58 560.13	52 939.17	559.496	248.907	0.184	0.466
	4	60 826.26	51 539.57	328.475	190.200	0.087	0.397
	5	57 671.38	50 921.76	357.716	273.608	0.114	0.499
	6	60 627.97	52 018.22	285.062	328.199	0.117	0.332
	7	57 109.97	49 754.08	527.511	220.205	0.099	0.489
	8	61 787.16	53 005.12	608.171	308.575	0.081	0.453
	9	58 200.52	48 186.65	909.538	270.176	0.098	0.603
	10	61 431.01	52 886.72	300.866	249.418	0.070	0.444
200 jobs	1	304 597.46	180 538.52	7958.944	779.658	0.131	0.418
	2	268 838.28	179 396.06	334.042	1399.587	0.126	0.253
	3	226 827.29	180 308.98	3900.617	574.250	0.317	0.210
	4	270 193.42	187 776.00	2393.788	922.474	0.089	0.038
	5	292 772.86	183 172.70	2846.380	213.395	0.008	0.227
	6	285 062.78	179 408.98	6500.727	801.019	0.094	0.176
	7	262 674.38	178 711.64	1940.751	389.683	0.051	0.009
	8	260 307.49	186 315.89	1156.127	202.413	0.591	0.203
	9	275 382.96	185 023.65	4800.215	1158.635	0.251	0.163
	10	222 254.75	180 514.74	5914.733	264.625	0.245	0.248
500 jobs	1	1 809 329.85	1 211 252.1	31 374.301	8416.726	0.037	0.258
	2	1 855 759.85	1 191 234.7	38 577.023	14 066.788	0.154	0.419
	3	1 845 380.06	1 237 259.0	28 851.034	10 306.685	0.179	0.415
	4	1 837 733.96	1 476 014.0	28 588.734	8482.296	0.435	0.062
	5	1 794 738.27	1 346 822.3	28 473.096	7857.409	0.576	0.133
	6	2 023 422.56	1 333 836.4	59 021.761	11 197.513	0.072	0.009
	7	2 298 323.68	1 325 310.8	84 795.500	3069.009	0.022	0.144
	8	1 699 541.12	1 337 581.1	16 254.984	10 266.789	0.038	0.016
	9	1 586 999.21	1 222 285.8	2715.093	6993.524	0.128	0.256
	10	1 952 172.80	1 414 805.6	54 789.678	5844.314	0.158	0.080
	Average	1 870 340.14	1 309 640.2	37 344.120	8650.105	0.180	0.179

In this part of the computational experiments, some large-size instances are generated in order to compare the performance of our proposed heuristic algorithm and GA. Using about 40 test problems, some useful comparisons were made to compare the quality of solutions considering indices MID, RAS and SAS. Table 3 represents the comparison results for $F = 3$, $m^f = (4, 3, 5)$ and $v^f \in [1, 1.2]$. The processing times of each instance are generated using a discrete Uniform distribution from 70 to 100.

TABLE 4. Kruskal–Wallis ANOVA table for methods with MID index.

Source	df	Sum of square	Mean square	χ^2	Prob $> \chi^2$
Method	1	2000	2000	3.7	0.0543
Error	78	40 660	521.282		
Total	79	42 660			

TABLE 5. Kruskal–Wallis ANOVA table for methods with SNS index.

Source	df	Sum of square	Mean square	χ^2	Prob $> \chi^2$
Method	1	8241.8	8241.8	15.26	9.3551e-005
Error	78	34 418.2	441.26		
Total	79	42 660			

TABLE 6. Kruskal–Wallis ANOVA table for methods with RAS index.

Source	df	Sum of square	Mean square	χ^2	Prob $> \chi^2$
Method	1	8425.5	8425.51	15.6	7.81085e-005
Error	78	34 232.5	438.88		
Total	79	42 658			

The preliminary results indicate a considerable potential to obtain good solutions through the implementation of the MSLS. From the results shown in Table 3, it is clear that a significant difference exists between MSLS and GA, especially when index MID is considered. The GA is capable of generating more diverse solutions, according to the SNS index. Also, it can be seen, MSLS and GA have similar results when index RAS is concerned. To verify the statistical validity of the results shown in Table 3 and confirm which the best algorithm between MSLS and GA is, a Kruskal–Wallis test as a non-parametric method has been performed in which the different algorithms as a factor and the response variable as an index value are considered. Note that, the parametric equivalent of the Kruskal–Wallis test is the one-way analysis of variance (ANOVA). A significant Kruskal–Wallis test indicates that at least one sample stochastically dominates one other sample. The obtained results are shown in Tables 4–6 and Figures 8–10.

From the data stored in Tables 5 and 6, it is clear that a significant difference exists between MSLS and GA. Hence, regarding the MID index, Table 4 shows there is no significant difference exists within algorithms at the 95% confidence level.

As it is also seen in Table 3, considering index MID, the efficiency of the algorithms becomes worse. However, this trend reverses concerning the second criterion, *i.e.*, regarding the SNS index, the spread of non-dominated solutions increases. Moreover, regarding the RAS index, again, the rate of simultaneous achievement to two objectives, improves. This table also shows our proposed algorithm can be applied in the instance with 500 jobs in ten factories. This matter reveals the importance of proposing a heuristic to obtain good solutions in a short computational time for large-size instances. Figures 11–13 show the mean plot and least significant difference (LSD) intervals at the 95% confidence level for the interaction between the type of algorithm and the number of jobs for three indices.

As shown in Figures 11 and 12, MID and SNS have an increasing trend when the number of jobs increases. From the information shown in Figure 13, it is clear that both algorithms have robust behavior when the number of jobs changes.

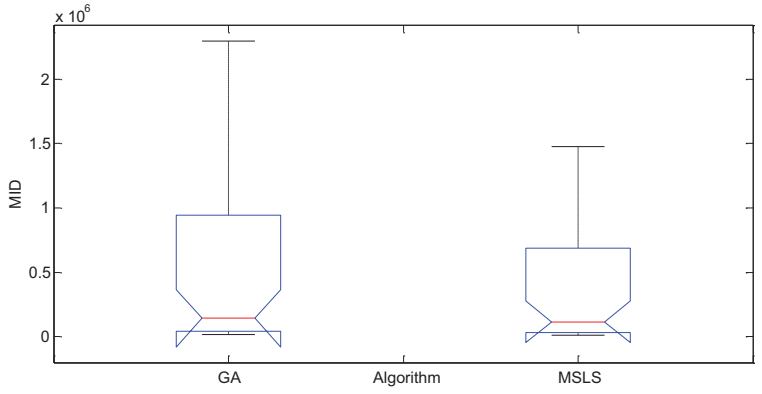

FIGURE 8. Kruskal–Wallis chart for methods with MID index.

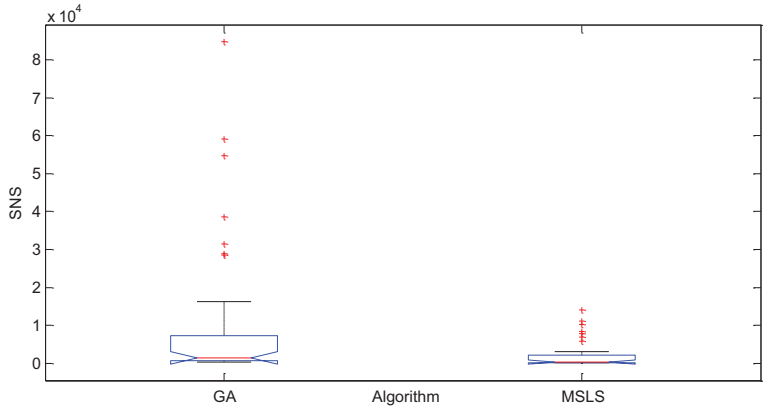

FIGURE 9. Kruskal–Wallis chart for methods with SNS index.

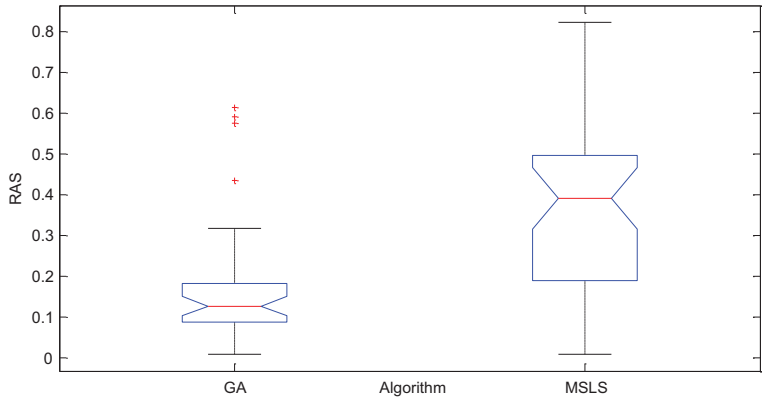

FIGURE 10. Kruskal–Wallis chart for methods with RAS index.

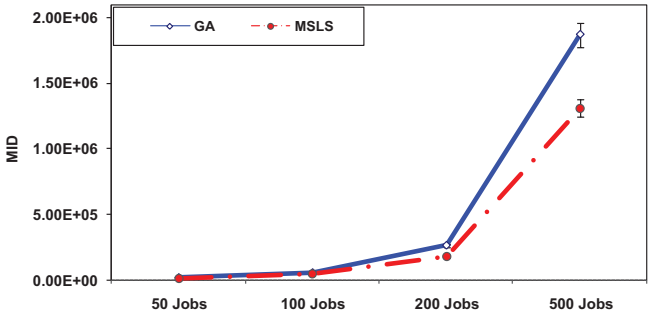

FIGURE 11. Plots of evaluation metric value for the interaction between the type of algorithm and number of jobs with index MID.

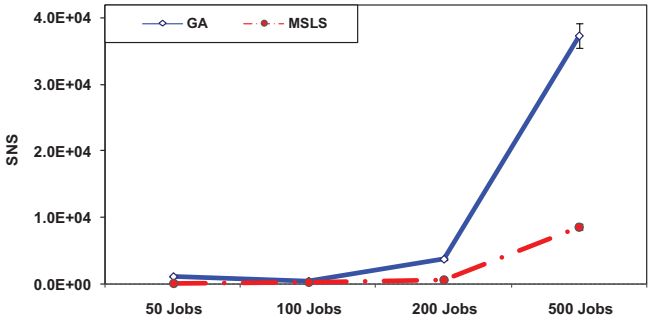

FIGURE 12. Plots of evaluation metric value for the interaction between the type of algorithm and number of jobs with index SNS.

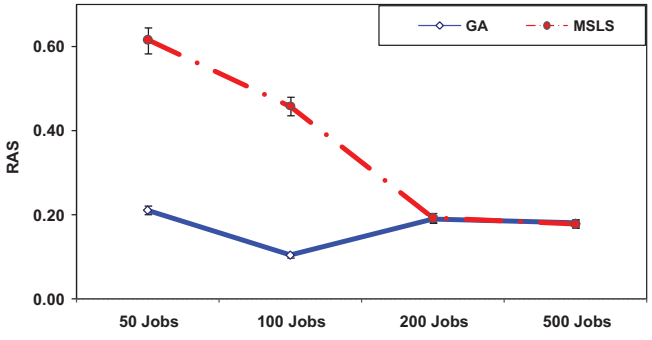

FIGURE 13. Plots of evaluation metric value for the interaction between the type of algorithm and number of jobs with index RAS.

7. Conclusions and future research

This paper considered a multiple objectives multi-factory scheduling problem with parallel machines. A mathematical model was presented for the problem and after proposing an appropriate multi-objective technique, it was implemented in CPLEX for test instances up to 15 jobs and three factories. To improve the obtained Pareto solutions, by using the relaxed problem, several sub-problems for each couple of Pareto solutions were

generated. The aim of solving these sub-problems is to find a new solution in the corresponding zones. Due to high complexity and considerable time to solve the proposed model exactly, an effective multi-objective heuristic approach, namely multi-structure local search (MSLS) which has six neighborhood structures, is designed. In the obtained Pareto solutions, after detecting the sparse part of them, a covering technique with the aim of improving the diversity and convergence of the MSLS results is proposed. For the real-life application, the large-size test benchmarks are generated. Results indicated better performance of MSLS in comparison with a genetic algorithm. Regarding the mean ideal distance index our proposed algorithm yields better solutions while the GA shows better performance when the spread of non-dominated solutions is concerned. The rate of simultaneous achievement to two objectives in the two algorithms (especially in large-size instances) is similar. Furthermore by increasing the number of jobs the algorithms lose their qualities considering the mean ideal distance and solutions spread indices.

The current paper ignores some realistic assumptions that can be studied in future research. Some of them are as follows:

- Here, it is assumed machines are continuously available, but in most real-life industries, a machine can be unavailable for many reasons. So, in future research, it can be assumed that the machines are not continuously available at all time in all factories.
- Also, in the current research, the assumption is that job information is available before the scheduling is started. But in real scheduling, such information may not be known beforehand and when a job is available, it must immediately be assigned to one of the factories before the next job becomes available.
- Moreover, practical production usually operates in stochastic events, such as random job arrivals, machine breakdowns and due dates. Stochastic multi-factory scheduling can be interesting for future research.
- Considering the production network with combined structures (*i.e.*, parallel structure combined with series structure) and network scheduling with dissimilar machine environments in each factory are also some open areas on generalizing the proposed problem.
- In this paper, some neighborhood structures were proposed which can be improved, or other neighborhood structures can be designed.
- Using other multi-objective techniques to solve the problem is our last suggestion for future research.

References

[1] A. Alarcon-Rodriguez, G. Ault and S. Galloway, Multi-objective planning of distributed energy resources: a review of the state-of-the-art. *Renewable Sustainable Energy Rev.* **14** (2010) 1353–1366.

[2] J.C. Beck and P. Refalo, A hybrid approach to scheduling with earliness and tardiness costs. *Ann. Oper. Res.* **118** (2003) 49–71.

[3] J. Behnamian and S.M.T. Fatemi Ghomi, The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machines. *Inf. Sci.* **219** (2013) 181–196.

[4] J. Behnamian, S.M.T. Fatemi Ghomi and M. Zandieh, A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Syst. App.* **36** (2009) 11057–11069.

[5] T.E. Carroll and D. Grosu, Selfish multi-user task scheduling. In: The Fifth International Symposium on Parallel and Distributed Computing (ISPDC '06), Timisoara (2006) 99–106.

[6] H.C. Chang and T.K. Liu, Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *J. Intell. Manuf.* **28** (2017) 1973–1986.

[7] V.A. Cicirello and S.F. Smith, Wasp-like agents for distributed factory coordination. *Auton. Agents Multi-Agent Syst.* **8** (2004) 237–266.

[8] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley and Sons, Delhi (2010).

[9] M.M. Dessouky, Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. *Comput. Ind. Eng.* **34** (1998) 793–806.

[10] G. Feng and H.C. Lau, Efficient algorithms for machine scheduling problems with earliness and tardiness penalties. *Ann. Oper. Res.* **159** (2008) 83–95.

[11] M. Gendreau and J.-Y. Potvin, editors. Handbook of Metaheuristics, 2nd edition. Vol. 272 of *International Series in Operations Research & Management Science*. Springer, New York, NY (2010).

[12] M.G. Gnonia, R. Iavagnilioa, G. Mossaa, G. Mummoloa and A. Di Leva, Production planning of a multi-site manufacturing system by hybrid modelling: a case study from the automotive industry. *Int. J. Prod. Econ.* **85** (2003) 251–262.

[13] J. Grobler, A.P. Engelbrecht, S. Kok and S. Yadavalli, Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time. *Ann. Oper. Res.* **180** (2010) 165–196.

[14] Ö. Hazır and S. Kedad-Sidhoum, Batch sizing and just-in-time scheduling with common due date. *Ann. Oper. Res.* **213** (2014) 187–202.

[15] T. İnkaya and M. Akansel, Coordinated scheduling of the transfer lots in an assembly-type supply chain: a genetic algorithm approach. *J. Intell. Manuf.* **28** (2017) 1005–1015.

[16] S.C.H. Leung, Y. Wu and K.K. Lai, Multi-site aggregate production planning with multiple objectives: a goal programming approach. *Prod. Planning Control* **14** (2003) 425–436.

[17] J. Lin, Z.J. Wang and X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm Evol. Comput.* **36** (2017) 124–135.

[18] T. Meng, Q.-K. Pan and L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint. *Knowl. Based Syst.* **184** (2019) 104894.

[19] M. Ramanauskas, D. Šešok, R. Belevičius, E. Kurilovas and S. Valentinavičius, Genetic algorithm with modified crossover for grillage optimization. *Int. J. Comput. Commun. Control* **12** (2017) 393–403.

[20] R. Ruiz and T. Stuetzle, An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *Eur. J. Oper. Res.* **187** (2008) 1143–1159.

[21] M. Sambasivan and S. Yahya, A Lagrangean-based heuristic for multi-plant, multi-item, multi-period capacitated lot-sizing problems with inter-plant transfers. *Comput. Oper. Res.* **32** (2005) 537–555.

[22] W. Shao, D. Pi and Z. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. *Knowl. Based Syst.* **137** (2017) 163–181.

[23] S. Terrazas-Moreno and I.E. Grossmann, A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chem. Eng. Sci.* **66** (2011) 4307–4318.

[24] C.H. Timpe and J. Kallrath, Optimal planning in large multi-site production networks. *Eur. J. Oper. Res.* **126** (2000) 422–435.

[25] F.M. Westfield, Marginal analysis, multi-plant firms, and business practice: an example. *Q. J. Econ.* **69** (1955) 253–268.

[26] J.F.H. Williams, Heuristic techniques for simultaneous scheduling of production and distribution in multi-echelon structures: theory and empirical comparisons. *Manage. Sci.* **27** (1981) 336–352.

[27] X. Wu, X. Liu and N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. *Memet. Comput.* **11** (2019) 335–355.

[28] G. Zhang and K. Xing, Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion. *Comput. Oper. Res.* **108** (2019) 33–43.