RAIRO-Oper. Res. 50 (2016) 587–609
DOI: 10.1051/ro/2015049

RAIRO Operations Research
www.rairo-ro.org

MULTI-OBJECTIVE OPTIMIZATION OF INTEGRATED LOT-SIZING AND SCHEDULING PROBLEM IN FLEXIBLE JOB SHOPS

Mohammad Rohaninejad[1], Rashed Sahraeian[1] and Behdin Vahedi Nouri[2]

Abstract. This paper investigates a particular integrated lotsizing and scheduling problem in a multi-level multi-product, multi-machine and flexible routes environment that is called flexible job shop problem (FJSP). The considered problem involves making simultaneous decision in sequencing operations, sizing lots and assigning machines to operations in order to optimize a multi-objective function including minimizing sum of the system costs, total machines workload and makespan, while a given demand is fulfilled without backlogging. Due to the complexity of the problem, a hybrid meta-heuristic based on a combination of Genetic algorithm and particle swarm optimization algorithm is developed to solve it. Additionally, the Taguchi method is employed to calibrate the influential parameters of the meta-heuristic and boost its capabilities. Finally, the performance of the proposed algorithm is compared with some well-known multi-objective algorithms such as NSGAII, SPEA2 and VEGA. Regarding to the computational results, the hybrid algorithm surpasses the other algorithms in the closeness of solutions to pareto optimal front and diversity criteria.

Mathematics Subject Classification. 90B30, 90B05, 68T20, 90B50.

Received March 31, 2015. Accepted October 21, 2015.

1. Introduction

Nowadays, due to intense competition among manufacturing companies in global market, efficient production planning systems are required for delivering products in time as well as minimizing total costs of production. Lot-sizing and scheduling are two major issues in medium-term and short-term production planning which are closely interrelated. In most cases, planers consider these issues hierarchically in which the lot-sizing problem is solved at first and then the result is applied for the scheduling problem, which would not lead to high quality solutions. Therefore, in order to obtain globally optimal solutions, integrated models of lot-sizing and scheduling must be regarded to contemplate the interrelationship between different levels of planning and make simultaneous decisions.

Since 1985 that Karmarkar and Schrage [18] introduced this point of view, a large number of researches have been carried out to formulate and solve integrated lot-sizing and scheduling problems (ILSP). The vast majority of these efforts have been focused on single machine configuration [11,20,34]. Only a few authors have considered

Keywords. Lot-sizing, scheduling, flexible job shop, genetic algorithm, particle swarm optimization algorithm, TOPSIS.

[1] Department of Industrial Engineering, Faculty of Engineering, Shahed University, Tehran, Iran. rohaninejad.sm@gmail.com
[2] Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran.

Article published by EDP Sciences

other general and elaborate configurations including parallel machines [4, 15], flow shop [25, 26], flexible flow shop [2, 22], and job shop.

In this paper the integrated lot-sizing and scheduling in a multi-level and multimachine environment problem that is called flexible job shop problem (ILSP-FJSP) is investigated. So far, a handful research has been conducted on integrated lot-sizing and scheduling in multi-level and multimachine environment such as job shop problem (ILSP-JSP), but to the best of the authors' knowledge, there is no attempt on ILSP-FJSP in the literature. Fandel and Stammen–Hegene [9] presented a mathematical model for ILSP-JSP in capacitated, dynamic and deterministic cases with minimizing the sum of the sequence-dependent setup costs, the storage costs, the production costs and the costs of maintaining the machines' setup conditions. Ozturk and Orneck [23] developed a MIP model for the problem with allowable backordering and production capacity constraint along with linked lots. Karimi–Nasab *et al.* [17] formulated a special kind of ILSP-JSP where the working speed of machines can be changed and produced items should be assembled together to make final products. Moreover, they developed a memetic algorithm to solve it. An iterative procedure was developed by Gómez Urrutia *et al.* [10] in a way that a lagrangian heuristic solves the lot-sizing problem with detailed capacity constraints for fixed sequence, then a tabu search algorithm provides a new sequence, which can help to obtain a better production plan. This procedure is repeated during a certain number of iterations. Deleplanque *et al.* [7] studied the multi-item, multi-plant lot-sizing problem with transfer costs and capacity constraints. They developed a decomposition framework based Lagrangean relaxation, into a master facility location problem and a slave minimal cost multi-commodity flow problem.

According to the literature two groups of performance measurement can be distinguished in the integrated lot-sizing scheduling problems. First group related to production system costs, like pure lot-sizing problems' objective function, including minimizing the total cost of production, holding, backorder, setup and overtime, which almost all authors considered it. Second group related to time performance measurements, like pure scheduling problems' objective functions, such as minimizing makespan [27], average weighted tardiness, number of tardy jobs, total setup time, total flow time and total idle time of machines [24]. In order to approach more realistic, comprehensive and applicable model, as well as achieving overall optimum solution, these two kinds of measurement must be simultaneously taken into account. As a result, in this paper for the first time, a multi-objective function is examined which is consist of (1) minimizing Sum of the holding, setup, production, and overtime costs, (2) total machines workload and (3) total completion time (makespan).

The integrated lot-sizing and scheduling problem is strongly NP-hard [16], to tackle its complexity several solution methods have been developed. Some of these methods are based on mathematical modeling such as Rolling-horizon and Fix-and-Relax heuristic [1, 4, 27], fix and optimize heuristic [31, 33], which decompose the model into a set of smaller MIP models, each of which having a small number of binary variables, and the decision variables are then iteratively determined. According to the complexity of the problem the use of meta-heuristic methods for problem solving enjoys a special status. These methods involve genetic algorithm (GA) [2, 14, 32], tabu search (TS) [2, 29], simulated annealing (SA) [25, 27], which successfully have been applied to ILSP. In this paper, a novel meta-heuristic approach is proposed for solving the problem which combines the GA with the particle swarm optimization algorithm (PSO). The PSO algorithm aggregate more closely around each optimum and is potentially more powerful in solving NP-hard problems. The PSO as a powerful optimization algorithm is widely applied in scheduling problems [3, 19, 36] and lot-sizing problems [8, 12] separately; the implementation of this algorithm in ILSP is rare.

2. Problem definition

As mentioned above, this paper focuses on the simultaneous multi objective optimization of lotsize and scheduling of operations in the flexible job shop problem with taking into account sequence-dependent setup times. The FJSP is an extension of the job shop problem (JSP) by assuming that for each given operation, there is at least one machine to perform it. As it is shown in Figure 1, job 1 is consist of 4 operations that 1st operation is allowed to be processed by machines 1 or 2, the 2nd and 4th operations can only be processed by

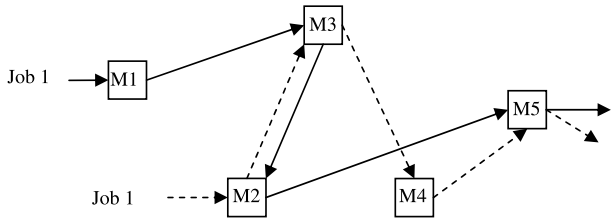

FIGURE 1. Schematic of the flexible job shop problem.

machine 3 and 5 respectively. Plus, for processing the 3rd operation, machines 2 and 4 are permitted. According to the above description it is clear that the FJSP is consists of two sub-problems; first, a routing sub-problem for assigning each operation to a machine from a set of capable machines; second, a scheduling sub-problem for sequencing the assigned operations on machines in order to obtain a feasible schedule. Therefore, the FJSP is one of the most difficult classes of scheduling problems.

In this paper the planning horizon consists of a finite number of periods and every period includes a specific number of time units. Machine capacities in each period are limited, and they are different from period to period. The machine capacity can be increased through overtime, and the overtime cost must be paid accordingly. The normal machine capacity and the permitted overtime capacity of each machine in every period are predetermined The other assumptions are made following:

1. All machines are continuously available from time zero.
2. A machine can only execute one operation at a time.
3. Every operation can be setup only once in each period.
4. Setup time is sequence-dependent.
5. Demand rates, production rates, setup times, setup costs, and inventory holding costs are deterministic and constant over the planning horizon.
6. External demands occur only for end operation of jobs.
7. Processing time rates are different for different products and different machines.
8. Shortages are not allowed.
9. In-process inventory is allowed: products may wait for their next machine to be free.
10. Production may be continued indefinitely without interruption.

In order to solve the problem with above conditions, it is necessary to set the problem variables in such a way that the objective functions of the problem become minimized. The objective functions of problem are:

f_1 : Sum of the holding, setup, production, and overtime costs.
f_2 : Total machines workload.
f_3 : Total completion time (makespan).

3. Proposed algorithm for solving the problem

In the solution space of ILSP-FJSP there are two types of the discrete and continuous variables. The discrete variables are consists of three sub-problems. (1) Determining the setup period(s) for each operation, (2) determining the assignment of operations to an appropriate machine, (3) determining the sequence of operations in each period. Also, determining the Lot quantity is continuous part of solution space. We developed a practical genetic algorithm for discrete space. With comparing PSO with other metaheuristic algorithms, it is shown the PSO is superior to them in terms of both efficiency and success rate for continues space. So for continuous space a special case of PSO algorithm is applied.

Furthermore, we have presented an effective *bottleneck shifting procedure*. This heuristic approach helps direct the search process to feasible points of the problem through the use of a neighbor search.

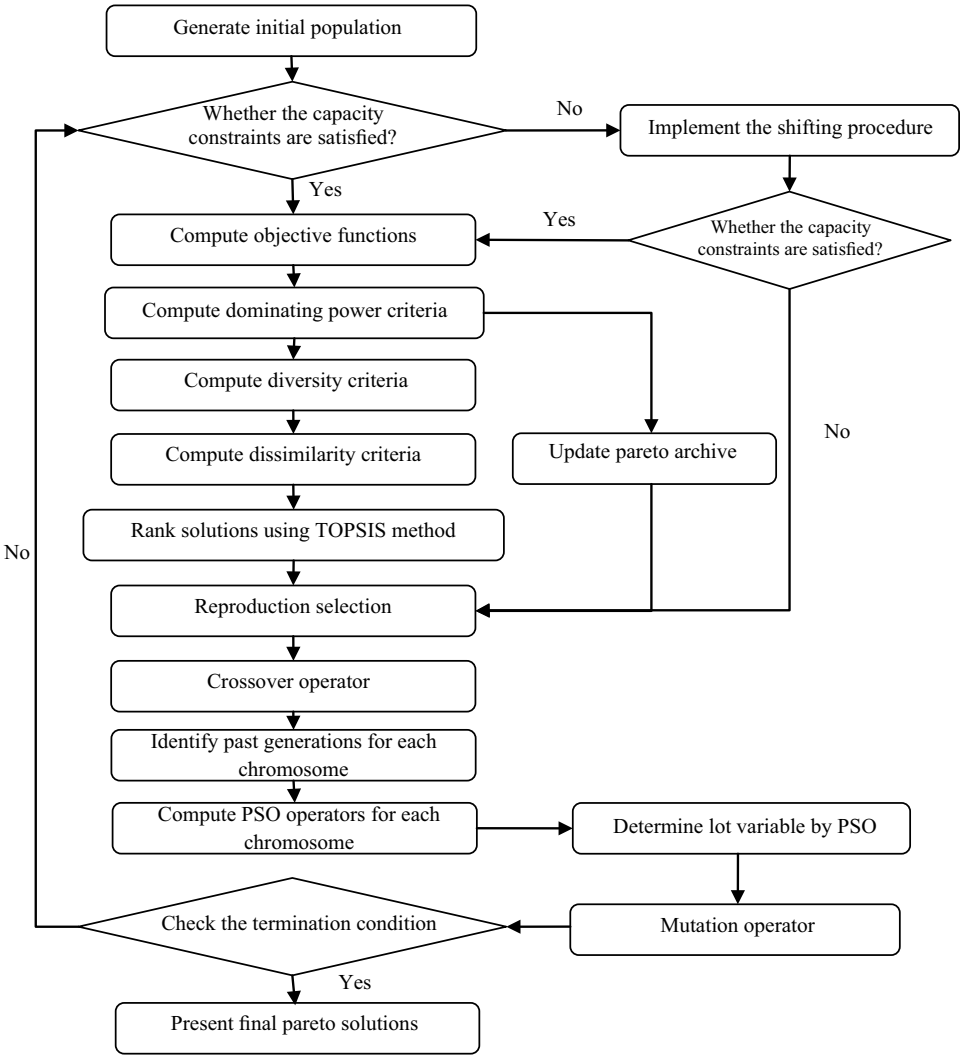

FIGURE 2. Schematic of the hybrid GA and PSO procedure.

Figure 2 shows the schematic of the hybrid GA and PSO procedure (HGAPSO).
In the following subsections, each step of the proposed approach is fully described.

3.1. Coding of chromosomes

To demonstrate the chromosomes of the genetic algorithm, a matrix with two rows and MNT columns is used in which M is the number of machines, N is the number of production positions for each machine in every period, and T is the number of periods throughout the planning horizon. In this matrix, the elements of the first row include three components (j, h, q) which indicate the quantities of item that are produced through operation $O_{j,h}$ are equal to q units. The arrangement of the matrix's first row components delineates the processing sequence of operations. Since each machine in every period has N production positions, the first MN arrays of first row indicate the sequences corresponding to the first period, and the tth MN arrays of first row indicate the sequences of MN positions corresponding to the tth period. In this problem, parameter N is

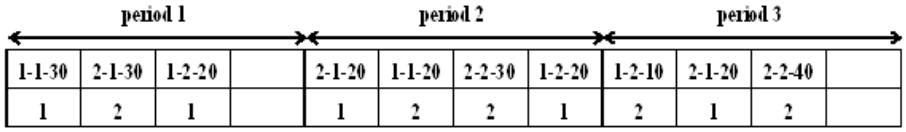

FIGURE 3. Manner of coding of chromosomes.

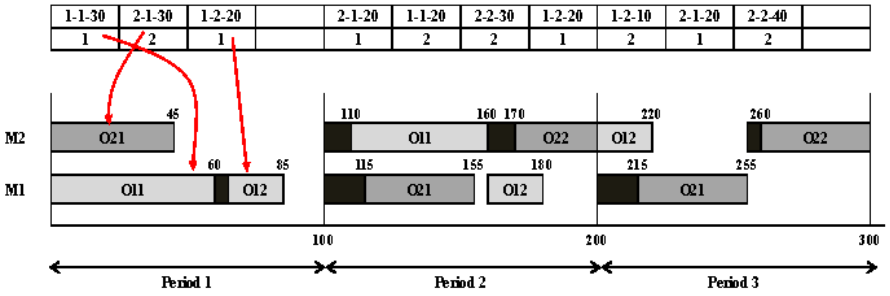

FIGURE 4. Manner of decoding of chromosomes.

determined in a way that all the operations get a production chance in a period. Also the second row of the matrix represents the assignments made to the first row elements. Figure 3 shows a chromosome for two jobs, each consisting of two operations which are supposed to be processed on two machines in a planning horizon that spans three periods.

3.2. Decoding of chromosomes

In the decoding of chromosomes, two factors should be considered: the start time for the processing of an operation, and the processing duration of that operation. The processing start time of an operation depends on when the machines related to that operation become available and when the needed inventory is provided to start processing the operation. Since both of the mentioned factors are necessary for the start of the operation, the start time for the operation is equal to the maximum of these two factors. Plus, the processing duration of operation $O_{j,h}$ will be equal to $a^m_{j,h} \cdot q$, if we assume that operation $O_{j,h}$ has been assigned to machine m. Figure 4 shows the manner of decoding of the chromosome.

In Figure 4, since the value of $a^1_{1,1}$ is assumed equal to 2, it is observed that in the first period, operation $O_{1,1}$ is processed for 60 time units. Furthermore, it can be seen that the values of $a^2_{1,1}$, $a^1_{2,1}$ and $a^2_{2,1}$ are assumed equal to 2.5, 2.0 and 1.5, respectively. In this figure, the dark sections indicate the sequence-dependent setup times. Moreover, in the third period of the planning horizon it is observed that the start time of operation $O_{2,2}$, despite the availability of machine 2 at time 220 is postponed to time 255, due to the insufficient quantity of inventory produced in operation $O_{2,1}$ (the coefficient of utilization of product from operation $O_{2,1}$ in the product of operation $O_{2,2}$ is assumed 1.0).

3.3. Generating the initial population of the algorithm

The genetic algorithm is one of the algorithms which are based on an initial population. To generate the initial population for the considered problem, it is necessary to determine the decision variables in such a way that, while facilitating the obtaining of the optimal (or nearly optimal) solution. In addition to cover the vast areas of the solution space, the suitable initial population helps search procedure to explore parts of the solution space in which high quality solutions are expected to be found. In view of the problem's decision variables, the generation of the initial population is based on the following four steps.

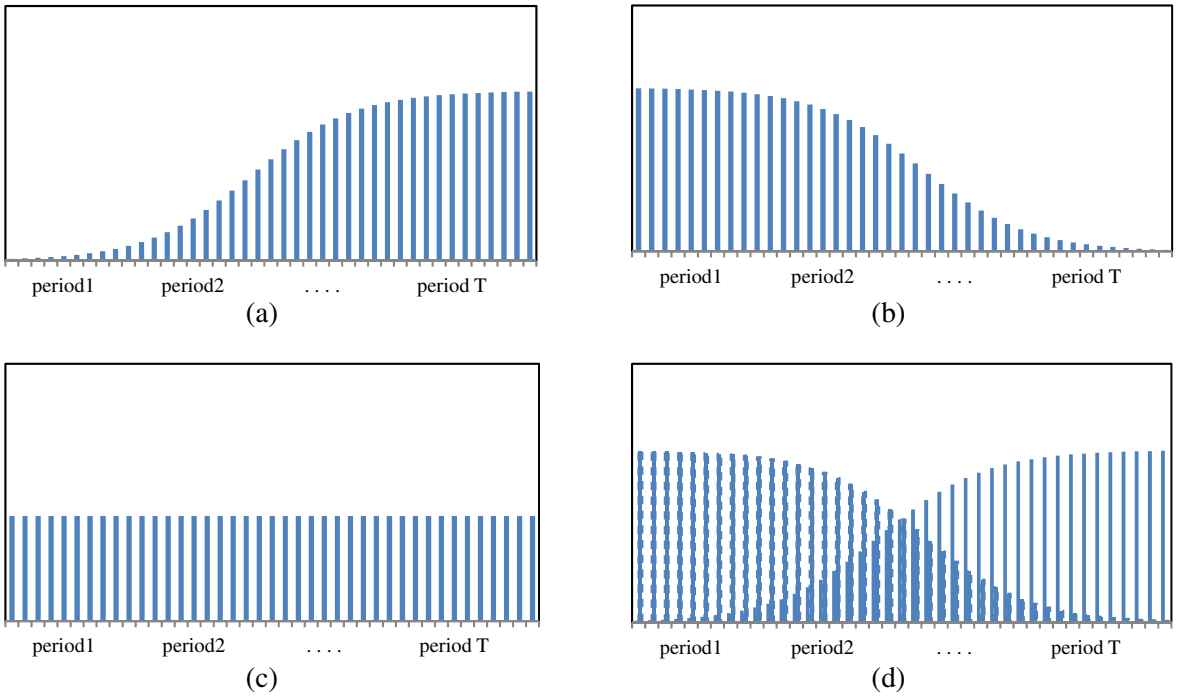

FIGURE 5. Four different types distributions for selection of production periods.

3.3.1. Determining the production periods for each item

In this step, the production periods of each item are determined on the basis of the following distribution:

$$P_1(x) = \frac{e^x/(100 + e^x)}{\sum\limits_{0<y<10} e^y/(100 + e^y)}, \tag{3.1}$$

$$P_2(x) = \frac{e^{(10-x)}/(100 + e^{(10-x)})}{\sum\limits_{0<y<10} e^{(10-y)}/(100 + e^{(10-y)})}, \tag{3.2}$$

$$P_3(x) = k \Big/ \sum_{0<y<10} k \qquad k \text{ is a constant.} \tag{3.3}$$

According to the above distributions, selection of production periods is done based on four different types that are shown in Figure 5. In type (a) which is based on $P_1(x)$, the latter periods have a greater chance for selection. Unlike type (a), in type (b) which is based on $P_2(x)$, the initial periods have a greater chance for selection. In type (c) that is based on $P_3(x)$, each period has an equal chance for selection and in type (d), the production periods for the first half of operations of each job are selected based on $P_1(x)$ and production periods for the second half of operations are selected based on $P_2(x)$.

In each distributions a particular period$t$is selected if $\frac{10(t-1)}{T} < x \leqslant \frac{10t}{T}$.

For the sake of preventing shortage, the selection of the first production period for each item is a function of the first period in which there is a demand for that item. In other words, the first production period of an item must definitely be less than or equal to the first period in which that item encounters a demand.

3.3.2. Determining the sequence of operations which are assigned to a period

To determine the sequences of operations which are assigned to a specific period, the following two methods are used:

- *Random sequence*: in this approach, one of the assigned operations is randomly selected and entered into the first unplanned sequence priority of that period.
- *Most Setup Remaining* (*MSR*): in this method, first the number of setups for every operation is specified. Then, each operation whose remaining setups (unplanned) are more than the others is selected and entered into the first unplanned sequence of that period. If there are more than one operation with the most remaining setups, one of them will be selected randomly.
- *Most Work Remaining* (*MWR*): in this method for each step the job that has the maximum remaining work (sum of processing time of remaining operations) is elected and its first unplanned operation is entered into the first unplanned sequence.

In the above methods, to avoid shortage, the first-time processing of an operation must never be planned after the first-time processing of its subsequent operation.

3.3.3. Determining the lot size values for each operation

It is important to determine the lot size values in a way that we don't run into shortage. The procedure to determine the lot size value of an operation is a forward procedure, and it starts from the first period and continues to the last. First, the total demands of an operation ($r_{j,h}$) between two consecutive setups s_1 and s_2 is calculated according to the following relations:

$$r_{j,h} = \sum_{s=s_1}^{s_2-1} \rho_{j,h} X^s_{j,h+1}. \tag{3.4}$$

Plus, for the last operation of each job (O_{j,h_j}):

$$r_{j,h_j} = \sum_{t=t_1}^{t_2-1} d_{j,t} \quad s_1 \in t_1 \text{ and } s_2 \in t_2. \tag{3.5}$$

In relation (3.5) $d_{j,t}$ is the external demand of job j in period t. Because of the dependency of the demand plan of an operation to the production plan of its subsequent operation, it is necessary to always start the lotsizing from the last operation of a job. After calculating parameter $r_{j,h}$, the lower limit of production in sequence s_1 is determined through the following relation:

$$l_{j,h} = \max(r_{j,h} - \Delta_{j,h}, 0) \qquad \Delta_{j,h} \geqslant 0, \tag{3.6}$$

where parameter $\Delta_{j,h}$ is the difference between total production and total consumption of the item which is produced in operation $O_{j,h}$ prior to sequence s_1.

Finally, the production quantity of operation $O_{j,h}$ item in sequence s_1 is equal to:

$$X^{s_1}_{j,h} = l_{j,h} + R\left(D_{j,h} - \Delta_{j,h} + \sum_{s=1}^{s_1-1} \rho_{j,h} X^s_{j,h+1} - l_{j,h}\right). \tag{3.7}$$

In relation (3.7), R is a random number between 0 and 1, and $D_{j,h}$ is the otal demand for the item of operation $O_{j,h}$ during the planning horizon.

3.3.4. Assigning machines to operations

To assign a machine to an operation, the following three methods can be used:

- *Random method*: in this method, one of the allowed machines is randomly assigned to each operation.
- *Earliest Time of machines Availability method* (*ETA*): in this approach, from the set of permitted machines for each operation, a machine which is available earlier than the other machines will be selected. Initially, a machine is selected randomly for the operation in the first sequence of each period. Then in each sequence following this assignment, the availability time of machines for the next operation is updated and ultimately, that operation is assigned to a machine which is available earlier than the other machines.
- *Shortest Setup Time method* (*SST*): in this method, from the set of permitted machines for an operation, a machine that has the shortest sequence-dependent setup time with respect to its last processed operation is selected.

3.4. Decision attributes and the TOPSIS method

The solution sets of the present population are ranked on the basis of the following three decision attributes:

1- *Dominating Power* (R_1):
This decision attributes that shows closeness of solutions to the optimal Pareto frontier is equal to:

$$R_1(x) = 1 + nq(x), \tag{3.8}$$

where, $nq(x)$ is the number of solution members of the current population which were defeated by the solution x.

2- *Diversity* (R_2):
Sum of Euclidean distances between a solution and Kth closest neighbors in the objective functions space is one of the common approaches for assessing diversity criteria in multi-objective problems. This attribute is calculated as follows.

$$R_2(x) = \sum_{x,y \in P} d_z(x,y), \tag{3.9}$$

where $d_z(x,y)$ is the normalized Euclidean distance between solution x and y which is calculated as follows:

$$d_z(x,y) = \sqrt{\sum_{j=1}^{k} \left(\frac{f_j(x) - f_j(y)}{f_j^{\max} - f_j^{\min}} \right)}. \tag{3.10}$$

The $f_j^{\max}$ and $f_j^{\min}$ are maximum and minimum values of f_j that obtained by algorithm until each step of solving process.

3- *Dissimilarity* (R_3):
Since the existence of diversity and variation among the problem's objectives does not guarantee the presence of diversity among the decision variables, the dissimilarity attribute is presented to provide the diversity conditions in the decision variables space of the problem which is calculated according to the following steps.

Step 1. *Similarity in production periods*:
First, number of similar production periods for each item in two different chromosomes x and y (q) is obtained and production periods similarity index ($UT_{x,y}$) is calculated as follow:

$$UT_{x,y} = \frac{q}{nT} \tag{3.11}$$

with comparing the two given chromosomes x and y in Figure 6, it can be seen that the seven genes in the two chromosomes have similar production periods. So, the $UT_{x,y}$ is equal to 7/12 ($n = 4$ and $T = 3$).

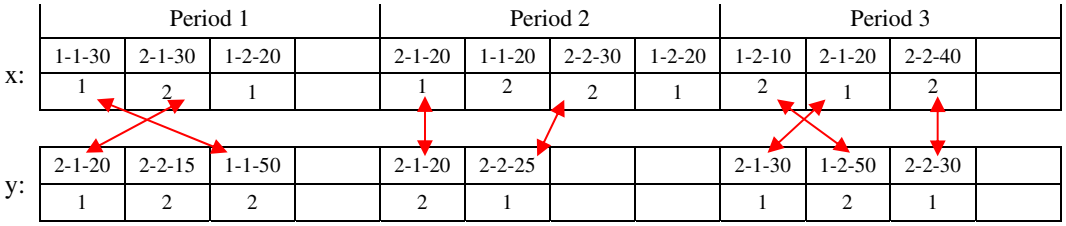

FIGURE 6. Similar genes in production periods between two chromosomes x and y.

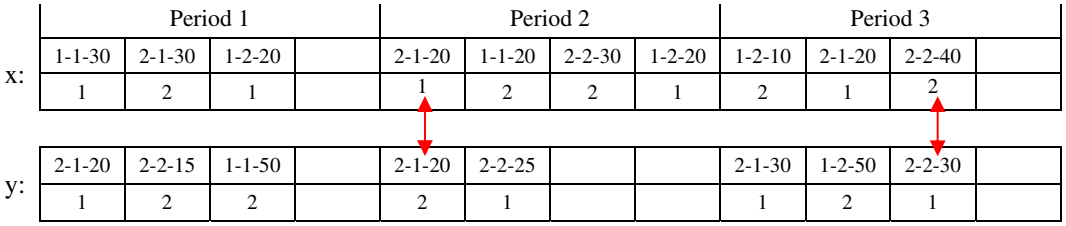

FIGURE 7. Similar genes in operations sequence between two chromosomes x and y.

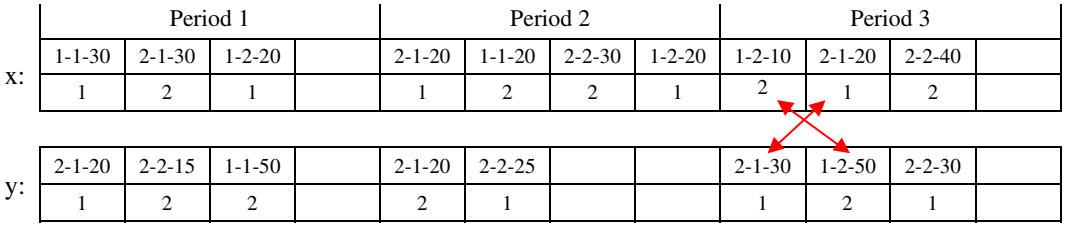

FIGURE 8. Similar genes in assignment between two chromosomes x and y.

Step 2. *Similarity in sequence of operations*:
If the value of q is greater than 0, then the sum of the number of same sequence for each operation (s) is obtained and $US_{x,y}$ is calculated as follow:

$$US_{x,y} = \frac{s}{q}. \tag{3.12}$$

In Figure 7, two operations $O_{2,1}$ in period 2 and $O_{2,2}$ in period 3 have same sequence in given chromosomes x and y. So, the $US_{x,y}$ is equal to 2/7.

Step 3. *Similarity in machine assignment*:
If the value of q is greater than 0, then the number of total same assignment for each operations in each period (a) is obtained and $UA_{x,y}$ is calculated as follow:

$$UA_{x,y} = \frac{a}{q}. \tag{3.13}$$

In Figure 8, operations $O_{1,2}$ and $O_{2,1}$ in period 3 of chromosomes x and y have same assignment. So, the $UA_{x,y}$ is equal to 2/7.

Step 4. *Similarity in lot variable*:
If $q > 0$, then the similarity index in lot variable ($UQ_{x,y}$) between two chromosomes x and y is calculated as blow:

$$UQ_{x,y} = \sum_{i=1,\ldots,q} \max\left\{\frac{\gamma_{share} - |q_{i,x} - q_{i,y}|}{\gamma_{share}}, 0\right\} \tag{3.14}$$

$|q_{i,x} - q_{i,y}|$ defines the difference in the lot variable quantities for an operation in same period between two chromosomes x and y while $q_{i,x}$ and $q_{i,y}$ are greater than 0. The γ_{share} is a basic amount which the difference between $q_{i,x}$ and $q_{i,y}$ is preferred to be greater than it.

According to Figure 6, the Similarity in lot variable index ($UQ_{x,y}$) for seven genes in the two chromosomes that have same production period is equal to 1.5 when γ_{share} is assumed to be 10.

$$UQ_{x,y} = 0 + 0 + 1 + 0.5 + 0 + 0 + 0 = 1.5.$$

Finally, each chromosome is compared with Kth closest neighbors in the objective functions space. Then the values of $UT_x = \Sigma_y UT_{x,y}$ and $US_x = \Sigma_y US_{x,y}$ and $UA_x = \Sigma_y UA_{x,y}$ and $UQ_x = \Sigma_y UQ_{x,y}$ are obtained where y is a member of Kth closest neighbors set for chromosome x. After normalizing the UT_x, US_x, UA_x and UQ_x the dissimilarity criteria $R_3(x)$ is calculated as follows:

$$R_3(x) = UT_x + US_x + UA_x + UQ_x. \tag{3.15}$$

Moreover, in each step, the current population solutions are ranked by using the TOPSIS method and the basis of above attributes. TOPSIS (technique for order preference by similarity to an ideal solution) method was presented by Hwang and Yoon [20]. The basic principle is that the chosen alternative should have the shortest distance from the ideal solution and the farthest distance from the negative-ideal solution. TOPSIS method ranks the preference order by calculating the relative closeness to the ideal solution.

The weight of the above attribute is dynamically modified throughout the implementation of the algorithm. Such that the weight of R_1, R_2 and R_3 in the beginning of the algorithm is equal to (0.5,0.3,0.2) and at the end of the algorithm is equal to (0.8,0.15,0.05). After applying the TOPSIS method, the solutions are organized in accordance with their ranks. Let, there are ϑ solutions, the solution with the highest value gets the rank number ϑ, and the one with the lowest value gets the rank number 1.

3.5. Selection for reproduction

After calculating the rank for each member solution of the current population, a tournament is used to select solutions from the population in order to produce the next generation. In this tournament, two solutions are randomly selected and a decision is made in the following manner:

- When comparing two feasible solutions, the one with greater rank value is selected.
- When comparing a feasible solution with an infeasible one, the feasible solution is always preferred and selected.
- When comparing two infeasible solutions, one of them is randomly selected.

3.6. Pareto archive and the evolution strategy

At each step, the dominating solutions of the current population are entered into the pareto archive. If the maximum capacity of the archive (U) is full and the solutions of the current population do not dominate any of the pareto archive solutions, then, after estimating the distance of each solution to maximum kth closest neighbors (for $U + b$ solutions that the b is the total number of candidate solutions for adding to archive), the b solutions which have the closest distance are removed from the archive. Assuming all objective functions are minimization, a feasible solution x dominate solution y, if and only if, $f_i(x) \leqslant f_i(y)$ for all objective functions (f_i is ith objective function) and $f_j(x) < f_j(y)$ for at least one objective function f_j. Here, elitism is expressed as a percent. For instance, elitism 10% means that in each step maximum 10% of next population will be selected from the current solutions in pareto archive.

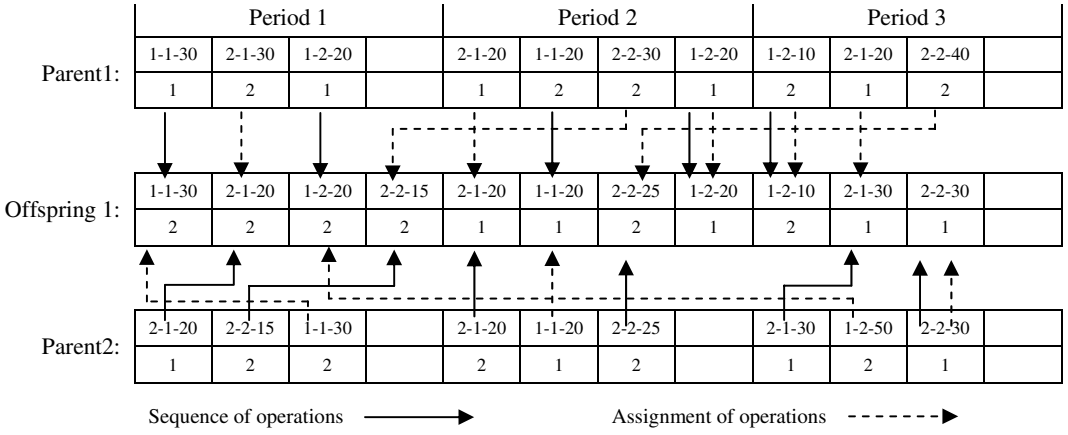

FIGURE 9. Manner of implementing the crossover operator.

3.7. Crossover operator

During the past decades, several crossover operators have been proposed for permutation representation, such as partial-mapped crossover, order crossover, cycle crossover, and so on. The employed crossover operator in this paper is an extension of the order crossover operator. This crossover operator, in addition to being simple, always produces a feasible solution if the chromosomes of the parent are feasible. In this operator, first, k jobs ($1 \leqslant k \leqslant J$) are randomly selected. Then for the generation of the first offspring, the positions of selected jobs are exactly copied into the first offspring. The sequences of the rest of the jobs are copied from the second parent into the first offspring, according to their sequence priorities.

The assignments of jobs are obtained as follows:

For operation $O_{j,h}$ (j is one of the selected jobs) let n be the number of setups for $O_{j,h}$ in parent 1 and m be the number of setup in parent 2. If $n \geqslant m$ then the assignment of m first setups are inherited from the second parent and assignment of $(n - m)$ remaining setups are inherited from the $(n - m)$ last setups of first parent. Also, if $n < m$ then the assignment of operation $O_{j,h}$ areinherited from the n first setups of second parent. This procedure is repeated for the remaining jobs in the reverse way and with the difference that n is belongs to parent 2 and m is belongs to parent 1.

Generating the second offspring is carried in the reverse way. Figure 9 shows an example of crossover operator acting on two assumed chromosomes. In this figure, job 1 has been selected for crossover operator implementation.

3.8. Mutation operator

The implementation of the mutation operator of the problem is based on re-planning k jobs ($1 \leqslant k \leqslant J$), which are chosen randomly. In this operator, the selected jobs are re-planned according to the explanations presented in "generating the initial population".

3.9. Determining the lot size values based on the particle swarm optimization algorithm

It is obvious that the execution of the crossover operator as described in Sections 3–7, confines the search in the lot-sizing decision variable space to the values determined in the initial population; thus, no search will be conducted in a vast range of these values. For this reason, we have employed an approach that combines the particle swarm optimization algorithm with the genetic algorithm in order to carry out a more extensive search in the lot-sizing decision variable space. In this approach, after implementing the crossover operator, the lot size

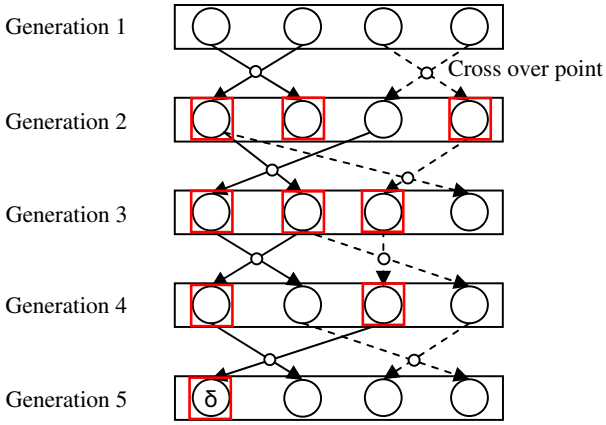

FIGURE 10. Manner of past generations identification of a chromosome with parameter $\lambda = 3$.

values of k jobs selected for the crossover procedure are determined in the offspring chromosome δ, as follows. In chromosome δ, we want to specify the lot size value of operation $O_{j,h}$, in period t and on machine m ($x_{j,h}^{m,t}$).

Step 1. *Determining the best visited position for the particle (pbest)*:
This component acts as a memory to preserve the position in which the particle has had a better status, and it is effective in the return of particle to its best previous position (like the inclination of people to return to a position or location that has satisfied them more in the past). In the proposed algorithm, the *pbest* parameter is determined with regards to the lot size values of past generations of a chromosome. The number of referrals to previous generations to obtain the *pbest* value is limited and specified by the predetermined parameter λ. Figure 10 shows the manner of past generations identification of offspring chromosome δ with $\lambda = 3$ and a population of 4 chromosomes.

After identifying the set of parents of chromosome δ for λ previous generations (E), the following pseudocode determines the *pbest*.

1. Best objective = a positive large number
2. $pbest = 0$
3. **For** each $i \in$ E
4. **If** $y_{j,h}^{m,t} = 1$ and objective function $(i) <$ Best objective **Then**
   Best objective = objective function (i)
   $pbest = x_{j,h}^{m,t}$
5. **End if**
6. **End for**
7. **IF** $pbest = 0$ **Then**
8. **For** each $i \in$ E
9. **If** $\exists$ a machine m' that $y_{j,h}^{m',t} = 1$ and objective function $(i) <$ Best objective **Then**
   Best objective = objective function (i)
   $pbest = x_{j,h}^{m',t}$
10. **End if**
11. **End for**
12. **End if**

In the above pseudo-code, the value of $y_{j,h}^{m,t}$ is 1 if operation $O_{j,h}$ is processed in period t and on machine m; otherwise, it will be 0.

Step 2. *Determining the global best (gbest) position explored by swarm*:
This social component is a like a criterion that particles try to attain. This component causes each particle to be drawn to the best position found by its neighbors. In the proposed algorithm, the *gbest* parameter equals the lot size value of operation $O_{j,h}$ in period $t$in the best solution found throughout the solving process.

Step 3. *Determining the lot size value*:
In this step, the following pseudo-code is implemented.

1. $V_{j,h}^{t(z+1)} = V_{j,h}^{t(z)} + c_1 R_1 \left(pbest - x_{j,h}^{m,t(z)}\right) + c_2 R_2 \left(gbest - x_{j,h}^{m,t(z)}\right)$
2. $x_{j,h}^{m,t(z+1)} = x_{j,h}^{m,t(z)} + V_{j,h}^{t(z+1)}$
3. **If** $x_{j,h}^{m,t(z+1)} > ux_{i,j}^{m,t}$ **Then**
   $x_{j,h}^{m,t(z+1)} = ux_{i,j}^{m,t}$
4. **Else if** $x_{j,h}^{m,t(z+1)} < lx_{i,j}^{m,t}$ **Then**
   $x_{j,h}^{m,t(z+1)} = lx_{i,j}^{m,t}$
5. **End if**

z	Indicates the iteration number of algorithm.
$x_{j,h}^{m,t}$	Lot size value of operation $O_{j,h}$, in period t and on machine m.
$ux_{i,j}^{m,t}$	Denotes the upper production limit of operation $O_{j,h}$ in period t and on machine m.
$lx_{i,j}^{m,t}$	Denotes the lower production limit of operation $O_{j,h}$ in period t and on machine m.
$V_{j,h}^{t}$	Initial velocity parameter for producing the item of operation $O_{j,h}$ in period t.
R_1 and R_2	Random number between 0 and 1.
c_1 and c_2	Acceleration constants for *pbest* and *gbest* respectively.

3.10. Bottleneck shifting procedure

Maes *et al.* [21] have proved that finding a feasible solution for the CLSP (Capacitated Lot-sizing problem) with setup time is NP-hard, definitely, a large portion of the optimal solution search process will be used up in the search of infeasible points of the solution space, and this will lower the quality of the algorithm's final solutions. For this reason, an effective *bottleneck shifting procedure* is presented. This approach is a modifying approach in contrast to strategies that eliminate or define a penalty function for infeasible solutions or applying a preventive tactic to avoid producing infeasible solutions. This approach involves the shifting of productions part of a period t to period $t-1$ (if machine capacity in period t has been violated). To implement this procedure, the following steps are taken:

Step 1. Identifying the last period in the planning horizon in which the machine or period capacity limitation has been violated: if capacity limitation is complied within all the periods, the shifting procedure is stopped. Also, if capacity limitation in period 1 has been violated, the shifting procedure is stopped without prosperity in generation of a feasible solution.

Step 2. *Calculation of the impermissibly used capacity in period t* (Q_t):
In this step, if the capacity limitation of machine m is violated in period t then:

$$Q_t = \sum_{j=1}^{J} \sum_{h=1}^{h_j} \sum_{r \in t} \left(a_{j,h}^{m} . X_{j,h}^{m,r} + \sum_{k=1}^{J} \sum_{l=1}^{h_k} \delta_{j,h,k,l}^{m} Z_{j,h,k,l}^{m,r} \right) - C_{m,t} + O_{m,t}. \tag{3.16}$$

And if the capacity limitation of period t is violated then:

$$Q_t = \max\left(\left(f^{m.r} - (t-1)\, TC \right) | r \in t \right) - PC. \tag{3.17}$$

TC: Total capacity or length of each period in the planning horizon.

Step 3. *Identifying the critical path of production scheduling*:
In this step, if the capacity limitation of machine m is violated in period t, the critical path includes all the operations which are processed on machine m plus the operations which cause the finish time of operations of machine m in period t to not be less than $(f^{m.r}|r = Nt)$. But if the capacity limitation of period t is violated, the critical path will consist of a set of operations which have no slack in the production scheduling of period t.

Step 4. *Shifting the first operation of the critical path*:
In this step, some production quantity of the first critical path operation (operation $O_{j,h}$) is shifted from period t to period $t-1$. It should be noted that the shifting must definitely be done from the first operation of the critical path; otherwise, encountering with shortage in period $t-1$ is possible. For this shifting process, assuming that operation $O_{j,h}$ is processed on machine m and $TP^t_{j,h}$ is equal to the total time of processing and setup of operation $O_{j,h}$ in period t the following procedure is followed:

- If $TP^t_{j,h} \leqslant Q_t$, all of the production of operation $O_{j,h}$ in period t $(qs_{j,h} = X^t_{j,h})$ is shifted to period $t-1$, and we return to step two.
- If $TP^t_{j,h} > Q_t$, part of the production of operation $O_{j,h}$ in period t $(qs_{j,h} = \min\{\frac{Q_t}{a^m_{j,h}}, X^t_{j,h}\})$ is shifted to period $t-1$.
- In order to schedule the quantities shifted to period $t-1$, the following pseudo-code is implemented:

1. **If** $\exists$ *a position* $r' \in t-1$ *and a machine* m' *that* $X^{r',m'}_{j,h} > 0$ **Then**
   $X^{r',m'}_{j,h} = X^{r',m'}_{j,h} + qs_{j,h}$
2. **Else if** machine m has at least an unplanned position in period $t-1$ **Then**
   Find first unplanned position $r' \in t-1$ on the machine m and $X^{r',m}_{j,h} = qs_{j,h}$
3. **Else**
   Select randomly a machine m' with unplanned positions
   Find first unplanned position r' on the machine m' and $X^{r',m'}_{j,h} = qs_{j,h}$
4. **End if**

Step 5. *Modifying the production sequence of items in period* $t-1$:
In some cases, as a result of shifting part of the production of operation $O_{j,h}$ from period t to period $t-1$, the inventory balance inside period $t-1$ may be lost for operation $O_{j,h}$; in other words, $I^{t-2}_{j,h-1} < \rho_{j,h-1} X^{t-1}_{j,h}$. In these situations, operation $O_{j,h-1}$ has certainly been scheduled in period $t-1$ and in the following sequence of operation $O_{j,h}$, and it is sufficient to exchange the sequences of the two operations with each other.

$I^t_{j,h}$ Inventory level of operation $O_{j,h}$ at the end of period t.

4. Computation results

In this section, the performance of the proposed algorithm is evaluated in comparison with the three evolutionary genetic algorithms; VEGA by Schaffer *et al.* [30], NSGA II by Deb *et al.* [6] and SPEA2 by Zitzler *et al.* [37], which have been customized for the ILSP-FJSP. These algorithms have been coded by the MATLAB 7.1 software and run on a PC with 2.66 GHz processor and 4 GB of RAM.

In general, two main objectives of multi-objective optimization algorithms are: (a) finding optimal pareto solutions or solutions close to it; and (b) finding solutions with good diversity and variety. In other words, a multi-objective optimization algorithm is considered as an efficient algorithm if it can satisfy both of the noted

objectives [5] Therefore, two evaluation metrics which are considered for determining the performance of the algorithms are as follows:

1- *The set coverage metric* (SC): for the evaluation the closeness to the pareto optimal solutions. For each algorithm, it computes the number of final solutions which have not been dominated by the other algorithms solutions, divided by the total number of solutions of that algorithm.
2- *The spacing and spread metric* (SS): metric for the evaluation of diversity in pareto solutions. Here, two indexes presented by [5] are used to present a new effective metric. The first index estimates the relative distance of consecutive solutions. This metric is computed by:

$$\sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(d_i-\bar{d}\right)^2}, \tag{4.1}$$

where, $|Q|$ is equal to the number of final pareto solutions of the algorithm and $\bar{d}=\frac{\sum_{i=1}^{|Q|}d_i}{|Q|}$ and $d_i=\min_{k\in Q,k\neq i}\sum_{m=1}^{M}\left|f_m^i-f_m^k\right|$. The above index measures the standard deviation of different quantities of d_i. When solutions are uniformly next to one another, the values of spacing metric will be small; therefore, the algorithm with smaller spacing will be a better algorithm. One of the shortcomings of this metric is its inability in evaluating the extent of spread of the solutions along the pareto optimal front. The second metric is a maximum spread evaluation metric for measuring the spread of solutions along the pareto optimal front. This metric, is equal to:

$$\sqrt{\sum_{m=1}^{M}\left(\max_{i=1}^{|Q|}f_m^i-\min_{i=1}^{|Q|}f_m^i\right)^2}. \tag{4.2}$$

Thus, the higher the value of the above metric, the more spread have the set of obtained solutions. However, this metric cannot correctly evaluate the spread of the intermediate solutions. Therefore, in order to simultaneously evaluate the uniformity and spread of the solutions, Rohaninejad *et al.* [28] proposed the metric by combining the two above metrics.

$$\sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(d_i-\bar{d}\right)^2}\Bigg/\sqrt{\sum_{m=1}^{M}\left(\max_{i=1}^{|Q|}f_m^i-\min_{i=1}^{|Q|}f_m^i\right)^2}. \tag{4.3}$$

The set of algorithm solutions with a smaller spacing and spread metric will be in a better situation compared to the other algorithms.

4.1. Generating random instances for the problem

Since there is no benchmark for the problem of simultaneous optimization of lot size and scheduling of jobs in a flexible job, it is necessary to generate random instances for the problem in order to investigate the quality of the results obtained through the proposed hybrid meta-heuristic algorithm. Table 1 illustrates the way these instances are produced.

Also, the demand quantities of every job are generated based on a normal distribution with the average $\frac{480.T.M}{6.O}$ and variance 20, and randomly assigned to the periods of the planning horizon (O denotes the total number of operations and M is the number of machines).

Ultimately, 25 instances with the above conditions are generated in small size, medium size and large size that each instance is labeled with (α:β:γ:δ), which respectively indicate the number of jobs, the total number of operations, the number of machines and the number of periods in the planning horizon.

TABLE 1. Manner of generating random instances.

Parameter	Notation	Generated by
Length of each period	PC	between [200–480] unit
Available Capacity of machine m in period t	$C_{m,t}$	uniform distribution between [0.65PC–PC]
Upper limit of machine m overtime in period t	$O_{m,t}$	$480-C_{m,t}$
Production costs	$pc_{m,t}$	uniform distribution between [0.2–1]
Overtime costs	$oc_{m,t}$	$1.5pc_{m,t}$
Setup costs	$sc^m_{j,h}$	uniform distribution between [50–200]
Holding costs	$hc^t_{j,h}$	uniform distribution between [0.5–2]
Fixed loss of time for production unit of items	$a^m_{j,h}$	uniform distribution between [0.5–7]
Sequence-dependent setup times	$\delta^m_{j,h,k,l}$	uniform distribution between [10–60]
Production coefficient	$\rho_{j,h}$	1

TABLE 2. Factors and their levels for HGAPSO algorithm.

Factor	Levels	Number of levels
Elitism rate (A)	{0.15 ; 0.3}	2
Pc (B)	{0.7 ; 0.8 ; 0.85 ; 0.9}	4
Pm (C)	{0.05 ; 0.1 ; 0.15 ; 0.2}	4
Initial velocity (D)	{20 ; 40 ;60 ; 80}	4
$c1$ (E)	{0.5 ; 1 ; 1.5 ;2}	4
$c2$ (F)	{0.5 ; 1 ; 1.5 ;2}	4

4.2. Parameters calibration

In this work, the Taguchi method is applied for calibration the parameters of the proposed hybrid algorithm. This method has been presented by Taguchi in early 1960s, and it can be used in the designing of processes. In the Taguchi method, the value of quality characteristics obtained from the experiments is converted to signal-to-noise (S/N ratio) and used in some of the subsequent analyses. The designing procedure of Taguchi can be described as follows [35]:

- The influences of the controllable factors are evaluated over the S/N ratio and the mean of response. In fact the appropriate experimental design is performed over S/N ratio and the mean of the considered characteristic.
- For each factor that has significant impact on the S/N ratio, the level that increases the S/N ratio is selected.
- Each factor that does not have any significant impact on S/N ratio, and has significant impact on the mean of responses (y), is considered as an adjustment factor, and the level where its mean of y is closer to the objective point, is selected.
- The factors with no significant impact either on S/N ratio or on mean of y, is regarded as economic factors, and levels that decrease the cost of production are selected.

Taguchi categorizes objective functions into three groups: "the smaller-the better", "the larger-the better", and the "nominal-is-best" [35]. In this paper, the S/N ratio for the "the larger-the better" characteristic is calculated by:

$$S/N \text{ ratio} = -10 \log \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{y_i^2} \right), \tag{4.4}$$

where j, y_i and n denote the trial number, response variable and the number of replications, respectively. After examining the HGAPSO algorithm, it seems 6 factors can have significant effects on the algorithms, respectively. These factors and their proper levels are illustrated in Table 2.

The Taguchi design of HGAPSO algorithm is $L_{32}(2^1, 4^5)$, respectively. The selected Taguchi design has 32 different combinations of parameter levels. For every trial, five random instances (two small size, two medium

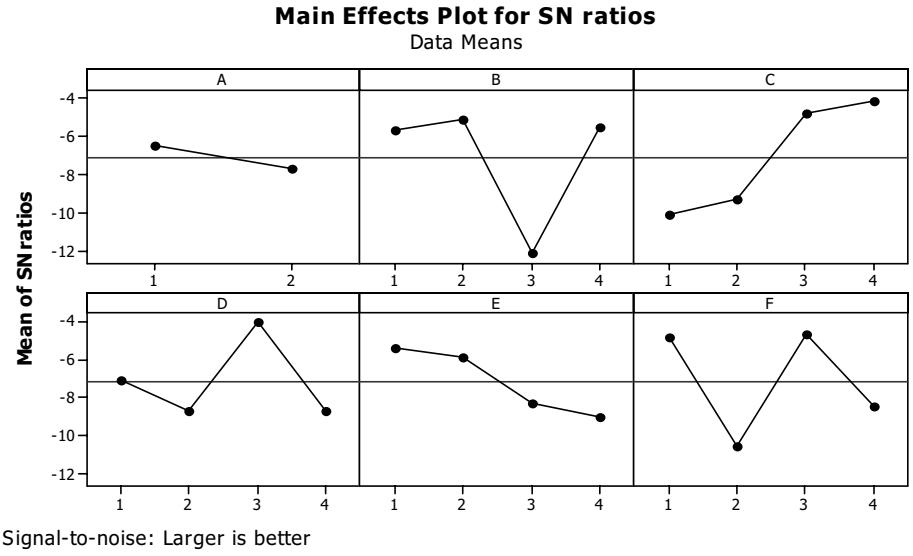

FIGURE 11. The mean S/N ratio plot for each level of the factors in HGAPSO algorithm.

TABLE 3. The parameters of the proposed algorithm.

Instance	Kth closest neighbors	Archive size	Population size	Number of iteration
Small size	4	15	20–25	50–100
Medium size	6	20	30–40	100–200
Large size	8	20	40	200–300

size and one large size instances) are considered and each of them is replicated five times in order to obtain more reliable results. In this paper, the mean RPD is used as a response variable and calculated as follows:

$$\text{mean}\,(RPD)_{ij} = \frac{\text{mean}_k(\text{objective}_{ijk}) - \min_i}{\min_i}, \tag{4.5}$$

where the objective is equal to:

$$\text{objective} = 0.6SC_{\text{normalized}} + 0.4\left(\frac{10}{SS}\right)_{\text{normalized}}, \tag{4.6}$$

where i, j and k indicate the instance index ($i = 1, \ldots, 5$), trial number ($j = 1, \ldots, 32$) and the replication number ($k = 1, \ldots, 5$), respectively. And $\min_i$ is the minimum value obtained for instance i. The mean RPD value is converted into the S/N ratio.

The mean S/N ratio is calculated for each level of control factors in HGAPSO algorithm and is plotted *versus* the control factors in Figure 11.

For the statistical significance test of factors on the S/N ratio, the analysis of variance (ANOVA) is used which shows the factors of algorithm have significant impact on the S/N ratio in HGAPSO algorithm. Therefore, these factors are considered as control factors and the levels with highest values of S/N ratio are selected as the optimal value for each of them. The optimal level for A, B, C, D, E, and F in HGAPSO algorithm are 0.15, 0.8, 0.2, 60, 0.5, and 0.5 respectively. Other parameters have been presented in Table 3.

TABLE 4. Fixed loss of time for production unit of items ($a^m_{j,h}$) – production cost for production unit of time ($pc^m_{j,h}$), setup cost ($sc^m_{j,h}$) and over time cost ($oc^m_{j,h}$) in (4:10:3:5) test case.

		$(a^m_{j,h} - pc^m_{j,h} - sc^m_{j,h} - oc^m_{j,h})$		
		Machine 1	Machine 2	Machine 3
Job1	Operation 1	1.0 – 1.7 – 135 – 2.5	1.0 – 3.3 – 172 – 4.9	1.1 – 1.1 – 126 – 1.6
	Operation 2	1.8 – 1.5 – 188 – 2.3	1.1 – 3.1 – 51 – 4.7	1.3 – 1.4 – 79 – 2.1
	Operation 3	1.5 – 3.1 – 121 – 4.6	–	1.5 – 1.5 – 194 – 2.2
Job2	Operation 1	2.2 – 2.4 – 130 – 3.6	0.9 – 2.1 – 114 – 3.1	2.5 – 2.9 – 114 – 4.4
	Operation 2	2.2 – 3.1 – 126 – 4.6	2.1 – 3.2 – 176 – 4.9	2.7 – 1.3 – 57 – 1.9
Job3	Operation 1	2.9 – 2.0 – 185 – 3.0	–	2.3 – 3.0 – 62 – 4.5
	Operation 2	2.9 – 2.0 – 103 – 3.0	2.1 – 2.4 – 113 – 3.6	1.3 – 1.5 – 189 – 2.2
	Operation 3	–	1.8 – 2.8 – 129 – 4.2	2.8 – 1.2 – 162 – 1.9
Job4	Operation 1	1.6 – 3.1 – 181 – 4.6	2.7 – 3.2 – 74 – 4.8	1.2 – 1.0 – 188 – 1.5
	Operation 2	2.3 – 1.9 – 63 – 2.8	1.1 – 1.0 – 159 – 1.5	–

TABLE 5. Holding cost in (4:10:3:5) test case.

		Period 1	Period 2	Period 3	Period 4	Period 5
Job1	Operation 1	1.1	1.3	1.9	1.3	1.3
	Operation 2	2.0	1.2	0.5	1.2	1.8
	Operation 3	1.4	0.8	1.8	0.9	1.6
Job2	Operation 1	1.7	1.8	1.4	0.5	0.6
	Operation 2	1.0	1.4	1.0	1.1	1.7
Job3	Operation 1	1.4	0.7	0.7	1.5	1.1
	Operation 2	1.2	1.0	1.3	0.8	1.9
	Operation 3	1.8	0.5	1.4	1.9	0.5
Job4	Operation 1	1.2	1.7	1.0	1.2	1.6
	Operation 2	1.7	1.1	1.9	1.1	0.9

4.3. Example

This section presents a numerical example (4:10:3:5) for better understanding of the problem. In (4:10:3:5) the external demand of jobs ($d_{j,t}$) and capacity of machines ($C_{m,t}$) are as follows:

$$
\begin{aligned}
&d_{1,5} = 130; \quad d_{2,4} = 100; \quad d_{3,2} = 60; \quad d_{3,3} = 30; \quad d_{4,4} = 20;\\
&C_{1,1} = 425; \quad C_{1,2} = 406; \quad C_{1,3} = 435; \quad C_{1,4} = 425; \quad C_{1,5} = 364\\
&C_{2,1} = 407; \quad C_{2,2} = 388; \quad C_{2,3} = 480; \quad C_{2,4} = 443; \quad C_{2,5} = 347.\\
&C_{3,1} = 370; \quad C_{3,2} = 376; \quad C_{3,3} = 380; \quad C_{3,4} = 302; \quad C_{3,5} = 463; \quad d_{4,5} = 40.
\end{aligned}
$$

Length of each period is equal to 480 time unit. Other information is shown in Tables 4 and 5.

Table 6 shows the patero solutions of the (4:10:3:5) test case which were obtained by solution methods.

4.4. Comparison and evaluation of the proposed solution methods

In this section the performance of VEGA, NSGA II, SPEA2 and HGAPSO for solving the considered problem are evaluated and compared. The obtained results of solving the small, medium and large size instances are listed in Tables 7 to 9, respectively.

By comparing the proposed algorithms, it is concluded that the HGAPSO algorithm has a more convincing performance than other algorithms based on the criteria of closeness of the solutions to the pareto optimal front and diversity.

TABLE 6. Present of pareto solution obtained by solution methods.

HGAPSO			VEGA			NSGA II			SPEA2		
f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
4079.6	1815.0	2157.0	4079.6	1815.0	2157.0	4093.0	1807.6	1687.0	3852.0	1859.0	2139.0
4628.5	1698.0	1305.0	4242.6	1817.0	1899.0	4218.0	1768.3	1508.0	4773.0	1685.0	1321.0
4826.8	1816.0	1290.0	4566.1	1827.0	1439.0	4387.0	1702.0	1295.0	4312.0	1720.9	1681.0
4999.2	1852.0	1185.0	4185.6	1785.0	2157.0	4844.0	1802.0	1197.0	3989.0	1971.6	1814.0
4196.2	1762.0	1687.0	4860.7	1728.0	1550.0	4714.0	1651.0	1504.0	4286.4	1750.6	1592.0
4208.0	1737.0	1538.0	4384.5	1687.0	1578.0	4773.0	1685.0	1321.0	4218.0	1768.3	1508.0
4093.0	1807.6	1687.0	4093.0	1807.6	1687.0	4826.8	1856.0	1229.0	4185.6	1785.0	2157.0
4298.0	1671.0	1418.0	4156.0	1850.0	1614.0	4516.4	1837.0	1262.0	4741.2	1765.1	1560.1
5011.6	1786.0	1201.0	4485.1	1967.0	1512.7				4999.2	1852.0	1185.0
5280.0	1814.0	1123.0	3813.0	1879.0	1961.0				5011.6	1786.0	1201.0
			5273.0	1725.6	1440.0				4826.8	1856.0	1229.0
									5332.0	1868.0	1155.0

TABLE 7. Details of computational results for small size instance.

Test problems	Time(s)				Num. of pareto solutions				SC Metric				SS Metric			
	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2
(2:4:2:2)	5.1	3.8	4.8	4.5	5	6	5	7	1.00	1.00	1.00	0.86	0.083	0.098	0.083	0.074
(2:4:2:3)	8.8	5.6	8.1	7.8	7	13	10	8	1.00	0.92	1.00	1.00	0.135	0.067	0.087	0.085
(2:6:2:2)	8.3	6.2	7.5	7.8	12	10	9	9	0.83	0.70	0.78	0.67	0.076	0.116	0.098	0.082
(2:6:2:3)	12.1	7.9	10.7	10.3	8	12	10	12	1.00	0.16	0.80	0.83	0.089	0.096	0.074	0.064
(3:6:2:4)	14.4	10.3	13.5	12.9	6	7	5	6	0.83	1.00	1.00	0.83	0.021	0.180	0.162	0.072
(3:8:2:2)	14.7	10.2	15.2	13.2	16	13	10	11	0.81	0.46	0.67	0.55	0.053	0.088	0.095	0.077
(3:8:2:3)	17.3	12.4	17.6	16.5	12	9	8	13	0.92	0.77	0.79	0.92	0.085	0.137	0.118	0.068
(4:10:2:2)	18.4	14.5	17.8	17.2	12	14	11	12	1.00	0.57	1.00	0.75	0.072	0.064	0.102	0.092
(3:8:3:4)	20.7	16.8	20.2	19.3	9	11	10	14	0.89	0.27	0.80	0.57	0.142	0.126	0.121	0.081
(4:8:3:5)	27.1	21.2	25.2	25.1	15	13	11	9	0.80	0.38	0.71	0.67	0.076	0.088	0.093	0.084

Figure 12 depicts the box plot based on the set coverage metric for each algorithm's solutions. This figure shows that the median of HGAPSO algorithm is equal to 0.89. In other word, 50% of the results are greater than or equal to this value. In addition, 75% of this algorithm's results are greater than or equal to 0.83 and 25% of results are equal to 1. The median of VEGA, NSGA II and SPEA2 algorithms are equal to 0.27, 0.79 and 0.68 respectively. Therefore, according to this plot, HGAPSO algorithm shows a better performance in set coverage metric rather than other algorithm.

Figure 13 illustrates the box plot based on the *spacing and spread* metric for each algorithm's solutions. According to this plot, the median of HGAPSO algorithm is equal to 0.073. In addition, 75% of this algorithm's results are less than or equal to 0.085 and 25% of results are less than or equal to 0.051. The median of VEGA, NSGA II and SPEA2 algorithms are equal to 0.104, 0.116 and 0.081 respectively. This figure reveals that HGAPSO algorithm excels other algorithm in *spacing and spread* metric. Outlier (*) is related to the results that are beyond the upper or lower whisker.

Figure 14 shows the number of dominated solutions in the population of each period during the 150 iterations of the HGAPSO for solving the instance with the population size of 40. The average numbers of dominated

TABLE 8. Details of computational results for medium size instance.

Test problems	Time(s)				Num. of pareto solutions				SC Metric				SS Metric			
	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2
(4:10:3:5)	32.8	28.4	30.2	31.4	10	11	8	12	1.00	0.45	1.00	0.67	0.051	0.78	0.098	0.082
(4:12:4:5)	54.4	41.2	48.4	50.3	7	12	8	13	0.57	0.33	0.63	0.62	0.073	0.133	0.145	0.060
(4:15:4:5)	67.9	52.6	65.3	62.9	11	9	5	6	0.91	0.00	0.80	0.67	0.045	0.128	0.108	0.084
(5:16:5:6)	109	88.7	107	106	10	9	11	13	0.80	0.00	0.55	0.54	0.084	0.104	0.121	0.078
(5:18:5:6)	126	104	123	124	20	7	14	20	0.85	0.00	1.00	0.80	0.070	0.127	0.166	0.081
(5:20:5:6)	144	112	141	139	20	14	9	18	0.85	0.07	0.89	0.61	0.052	0.096	0.184	0.084
(6:22:5:6)	181	134	176	170	8	13	11	15	1.00	0.00	0.64	0.73	0.114	0.132	0.120	0.095
(6:24:5:6)	244	195	238	235	19	20	15	17	0.84	0.20	0.93	0.59	0.60	0.98	0.116	0.107
(8:30:6:6)	416	375	408	396	13	18	14	16	0.61	0.05	0.57	0.38	0.116	0.164	0.214	0.087
(9:35:6:8)	678	535	637	612	20	13	12	15	0.90	0.38	0.75	0.53	0.068	0.101	0.154	0.114

TABLE 9. Details of computational results for large size instance.

Test problems	Time(s)				Num. of pareto solutions				SC Metric				SS Metric			
	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2	HGAPSO	VEGA	NSGA II	SPEA2
(10:40:6:10)	994	741	854	821	20	16	12	18	0.90	0.43	0.67	0.67	0.086	0.132	0.128	0.105
(12:45:8:10)	1440	1114	1320	1276	14	19	15	20	0.86	0.00	0.60	0.45	0.49	0.062	0.091	0.064
(12:50:8:12)	248	1715	1885	1830	19	19	16	20	1.00	0.00	0.81	0.60	0.032	0.111	0.133	0.056
(14:60:8:12)	2951	2486	2621	2678	18	15	20	18	0.89	0.13	0.75	0.83	0.051	0.128	0.107	0.074
(14:70:8:12)	3600	3522	3600	3600	20	20	18	20	1.00	0.00	0.78	0.70	0.083	0.084	0.112	0.061

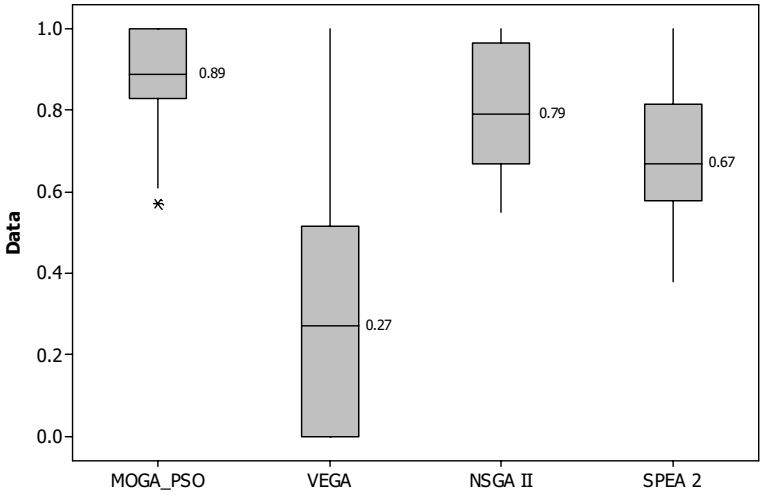

FIGURE 12. Box plot of set coverage metric for each algorithm with 95% confidence interval.

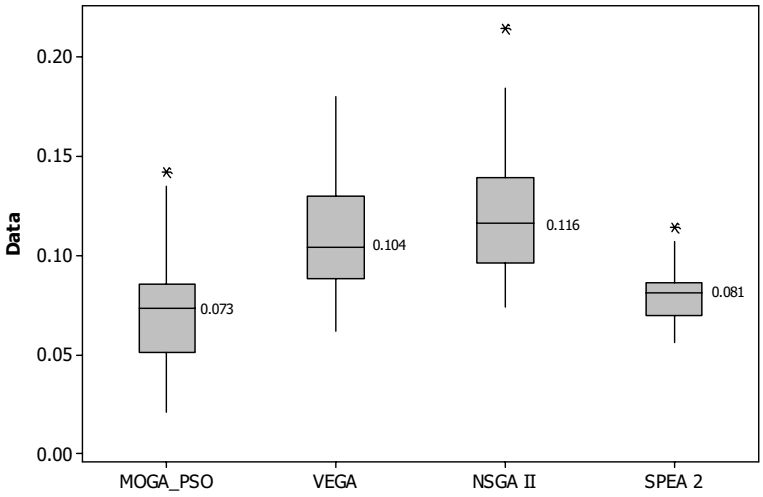

FIGURE 13. Box plot of spacing and spread metric for each algorithm with 95% confidence interval.

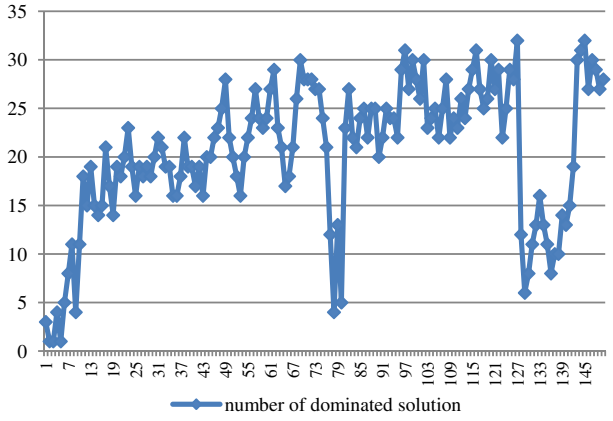

FIGURE 14. The number of dominated solutions in the population during the 150 iterations.

solutions are 18.6 and 22.3 during the first and second 75 iterations, respectively. Thus, this figure demonstrates the improvement of the results of the algorithm during the searching process.

5. Conclusion and future research directions

This paper dealt with multi-objective, simultaneous lot-sizing and scheduling problem in flexible job shop configuration under the machine capacity constraints and with the objectives of sum of the system costs, total machines workload and makespan. In order to solve this problem, a hybrid algorithm based on GA and PSO algorithms and TOPSIS method (HGAPSO) has been developed. The performance of the proposed algorithms was evaluated in comparison with three well-known evolutionary genetic algorithms including VEGA, NSGA II, and SPEA2. The computational results demonstrated that HGAPSO has significant advantage, particularly in the criteria of closeness to the pareto optimal front and diversity over other algorithms. Considering the problem with other common objective functions associated with the lot-sizing and scheduling problems, and devising more effective solution methods by means of other multi-attribute decision making approaches are suggested

for future extension of this research. In addition, adapting the procedure of this algorithm for solving other problems with both discrete and continues variables can be promising and advisable.

References

[1] N. Absi and S. Kedad-Sidhoum, MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. *RAIRO: OR* **41** (2007) 171–192.

[2] B. Akrami, B. Karimi and Moattar S.M. Hosseini, Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: The finite horizon case. *Appl. Math. Comput.* **183** (2006) 634–645.

[3] S. Arabameri and N. Salmasi, Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Comput. Ind. Eng.* **64** (2013) 902–916.

[4] P. Beraldi, G. Ghiani, A. Grieco and E. Guerriero, Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Comput. Oper. Res.* **35** (2008) 3644–3656.

[5] K. Deb, Multi-objective Optimization Using Evolutionary Algorithms. Vol. 2012. John Wiley & Sons, Chichester (2001).

[6] K. Deb, A. Pratap, S. Agarwal and T.A.M.T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6** (2002) 182–197.

[7] S. Deleplanque, S. Kedad-Sidhoum and A. Quilliot, Lagrangean Heuristic for a Multi–Plant Lot–Sizing Problem with Transfer and Storage Capacities. *RAIRO: OR* **47** (2013) 429–443.

[8] C.Y. Dye and L.Y. Ouyang, A particle swarm optimization for solving joint pricing and lot-sizing problem with fluctuating demand and trade credit financing. *Comput. Ind. Eng.* **60** (2011) 127–137.

[9] G. Fandel and C. Stammen–Hegene, Simultaneous lot sizing and scheduling for multi-product multi-level production. *Int. J. Prod. Econ.* **104** (2006) 308–316.

[10] E.D. Gómez Urrutia, R. Aggoune and S. Dauzère-Pérès, Solving the integrated lot-sizing and job-shop scheduling problem. *Int. J. Prod. Res.* **52** (2014) 5236–5254.

[11] K. Haase and A. Kimms, Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *Int. J. Prod. Econ.* **66** (2000) 159–169.

[12] Y. Han, J. Tang, I. Kaku and L. Mu, Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight. *Comput. Math. Appl.* **57** (2009) 1748–1755.

[13] C.L. Hwang and K. Yoon, Multiple Attributes Decision Making. *Lect. Notes Econ. Math. Syst.* Springer, Heidelberg (1981).

[14] M. Jenabi, S.M.T. FatemiGhomi, S.A. Torabi and B. Karimi, Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines. *Appl. Math. Comput.* **186** (2007) 230–245.

[15] W. Kaczmarczyk, Proportional lot-sizing and scheduling problem with identical parallel machines. *Int. J. Prod. Res.* **49** (2011) 2605–2623.

[16] B. Karimi, S.M.T. Fatemi Ghomi and J.M. Wilson, The capacitated lot sizing problem: a review of models and algorithms. *Omega* **31** (2003) 365–378.

[17] M. Karimi-Nasab, S.M. Seyedhoseini, M. Modarres and M. Heidari, Multi-period lot sizing and job shop scheduling with compressible process times for multilevel product structures. *Int. J. Prod. Res.* **51** (2013) 6229–6246.

[18] U.S. Karmarkar and L. Schrage, The deterministic dynamic product cycling problem. *Oper. Res.* **33** (1985) 326–345.

[19] A.H. Kashan and B. Karimi, A discrete particle swarm optimization algorithm for scheduling parallel machines. *Comput. Ind. Eng.* **56** (2009) 216–223.

[20] A. Kovács, K.N. Brown and S.A. Tarim, An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups. *Int. J. Prod. Econ.* **118** (2009) 282–291.

[21] J. Maes, J.O. McClain and L.N. Van Wassenhove, Multilevel capacitated lot-sizing complexity and LP-based heuristics. *Eur. J. Oper. Res.* **53** (1991) 131–148.

[22] M. Mahdieh, M. Bijari and A. Clark, Simultaneous Lot Sizing and Scheduling in a Flexible Flow Line. *J. Ind. Syst. Eng.* **5** (2011) 107–119.

[23] C. Oztürk and A.M. Ornek, Capacitated lot sizing with linked lots for general product structures in job shops. *Comput. Ind. Eng.* **58** (2010) 151–164.

[24] S. Petrovic, C. Fayad, D. Petrovic, E. Burke and G. Kendall, Fuzzy job shop scheduling with lot-sizing. *Ann. Oper. Res.* **159** (2008) 275–292.

[25] R. Ramezanian and M. Saidi-Mehrabad, Hybrid simulated annealing and MIP-based heuristics for stochastic lot-sizing and scheduling problem in capacitated multi-stage production system. *Appl. Math. Model.* **37** (2013) 5134–5147.

[26] R. Ramezanian, M. Saidi-Mehrabad and P. Fattahi, MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints. *J. Manufact. Syst.* **32** (2013) 392–401.

[27] R. Ramezanian, M. Saidi-Mehrabad and P. Fattahi, Integrated lot-sizing and scheduling with overlapping for multi-level capacitated production system. *Int. J. Comput. Integr. Manufact.* **26** (2013) 681–695.

[28] M. Rohaninejad, A. Kheirkhah, P. Fattahi and B. Vahedi-Nouri, A hybrid multi-objective genetic algorithm based on the ELECTRE method for a capacitated flexible job shop scheduling problem. *Int. J. Adv. Manufact. Technol.* (2014) 1–16.

[29] M. Rohaninejad, A.S. Kheirkhah, B. Vahedi Nouri and P. Fattahi, Two hybrid tabu search–firefly algorithms for the capacitated job shop scheduling problem with sequence-dependent setup cost. *Int. J. Comput. Integr. Manufact.* **28** (2015) 470–487.

[30] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. of the 1st international Conference on Genetic Algorithms*. L. Erlbaum Associates Inc. (1985) 93–100.

[31] F. Seeanner, B. Almada-Lobo and H. Meyr, Combining the principles of variable neighborhood decomposition search and the fix & optimize heuristic to solve multi-level lot-sizing and scheduling problems. *Comput. Oper. Res.* **40** (2013) 303–317.

[32] R. Sikora, A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. *Comput. Ind. Eng.* **30** (1996) 969–981.

[33] H. Stadtler, Multi-level single machine lot-sizing and scheduling with zero lead times. *Eur. J. Oper. Res.*, **209** (2011) 241–252.

[34] A. Weidenhiller and H. Jodlbauer, Equivalence classes of problem instances for a continuous-time lot sizing and scheduling problem. *Eur. J. Oper. Res.* **199** (2009) 139–149.

[35] C.J. Wu and M.S. Hamada, Experiments: Planning, Analysis, and Optimization. Wiley (2011).

[36] F. Zhao, J. Tang, J. Wang and J. Jonrinaldi, An improved particle swarm optimisation with a linearly decreasing disturbance term for flow shop scheduling with limited buffers. *Int. J. Comput. Integr. Manufact.* **27** (2014) 488–499.

[37] E. Zitzler, M. Laumanns and L. Theile, *SPEA2: Improving the strength Pareto evolutionary algorithm*. Technical Report (2001).