Iranian Journal of Operations Research
Vol. 3, No. 1, 2012, pp. 61-88

A Multi-Objective Particle Swarm Optimization Algorithm for a Possibilistic Open Shop Problem to Minimize Weighted Mean Tardiness and Weighted Mean Completion Times

S. Noori Darvish[1], R. Tavakkoli-Moghaddam[2,*], N. Javadian[3]

We consider an open shop scheduling problem. At first, a bi-objective possibilistic mixed-integer programming formulation is developed. The inherent uncertainty in processing times and due dates as fuzzy parameters, machine-dependent setup times and removal times are the special features of this model. The considered bi-objectives are to minimize the weighted mean tardiness and weighted mean completion times. After converting the original formulation into a single-objective crisp one by using an interactive approach and obtaining the Pareto-optimal solutions for small-sized instances, an efficient multi-objective particle swarm optimization (MOPSO) is proposed in order to achieve a good approximate Pareto-optimal set for medium and large-sized examples. This algorithm exploits new selection regimes of the literature for the global best and personal best. Furthermore, a modified decoding scheme is designed to reduce the search area in the solution space, and a local search algorithm is proposed to generate initial particle positions. Finally, the efficiency of the proposed MOPSO (PMOPSO) is shown by comparing with the common MOPSO (CMOPSO) by the use of the design of experiments (DOE) based on three comparison metrics.

Keywords: *Open shop scheduling, Weighted mean tardiness, Weighted mean completion times, Possibilistic programming, Particle swarm optimization.*

Manuscript received on 21/12/2010 and accepted for publication on 12/06/2011.

1. Introduction

Scheduling consists of an assignment problem and a sequencing problem to produce goods and provide services in a system. Among the several kinds of work environments, open shop as an interesting and universal problem seems to have received less attention from researchers and practitioners. Some applications of open shop scheduling problems (OSSP) are in testing, maintenance and teacher-class timetabling problems. The OSSPs are different from job shop and flow shop problems, because there is no precedence constraint between the operations of each job. It means the processing order of operations is immaterial. Thus, the solution space of OSSPs is much larger than other shop scheduling problems. This problem is known to be NP-hard. Therefore, using a proper meta-heuristic algorithm is necessary to achieve optimal or near-optimal solutions for medium and large-sized problems. Matta [18] developed two original mixed-integer programming (MIP) models (i.e., time-based model and sequence-based model) for the proportionate multiprocessor open shop scheduling problem and proposed a genetic algorithm (GA) to schedule the shop with the objective of minimizing the makespan.

* Corresponding Author.
[1] Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran.
Email: snooridarvish@gmail.com
[2] Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran.
Email: tavakoli@ut.ac.ir
[3] Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran.
Email: nijavadian@ustmb.ac.ir

Sha and Hsu [26] have proposed a new particle swarm optimization (PSO) for the OSSP to minimize makespan. In a comparison with the original algorithm, they have modified the particle position representation and the particle movement. They have hybridized their PSO with beam search. The computational results have shown that their PSO has found many new best solutions of the unsolved problems. Andresen et al. [2] have proposed a simulated annealing (SA) and a GA for the OSSP to minimize the sum of completion times or mean flow time. They have suggested several neighborhoods and used them to test SA, and proposed new genetic operators based on the representation of a solution for GA. Blum [4] has proposed a hybridizing ant colony optimization with beam search for OSSPs to minimize makespan. Blum and Samples [5] have considered group shop scheduling problems including the OSSP and the job shop scheduling problem as special cases to minimize makespan. They have proposed a neighborhood structure for this problem and developed an ant colony optimization (ACO) method that uses strong non-delay guidance for constructing solutions and employs black-box local search procedures to improve the constructed solutions.

Puente et al. [22] have proposed a method combining heuristic rules and GAs to solve an OSSP to minimize makespan. They have considered several dispatching rules taken from the literature that produce semi-optimal solutions in a polynomial time. They have designed probabilistic algorithms to generate heuristic chromosomes that are inserted in the initial population of a conventional GA. Liaw [15] has proposed a hybrid GA for the OSSP to minimize makespan. This hybrid algorithm incorporates a local improvement procedure based on tabu search (TS) into a basic GA. Prins [21] has developed competitive GAs for OSSP to minimize makespan. He has shown that even the simple and fast version of this algorithm can compete with the best known heuristics and meta-heuristics because of two key-features, namely, a population, in which each individual has a distinct makespan, and a special procedure that reorders every new chromosome. Seraj and Tavakkoli-Moghaddam [25] have proposed a TS to solve a new bi-objective mixed-integer mathematical programming model for an OSSP. This model minimizes the mean tardiness and the mean completion time.

In the real world, there is an inherent uncertainty in data and inputs of any systems. The data are articulated by linguistic terms such as around, about, less than, more than, etc. This kind of data named fuzzy/possibility data. In mathematical programming, when the input data are fuzzy or possibility, the mathematical model is transformed into a fuzzy or a possibilistic program. Torabi and Hassini [28] have designed a new multi-objective possibilistic mixed integer linear program model for supply chain master planning. After use of appropriate strategies to convert this possibilistic model into an auxiliary crisp multi-objective linear programming (MOLP) model, they have proposed a novel interactive fuzzy programming approach to solve this MOLP model and obtain a preferred compromise solution. Jiménez and Bilbao [10] have addressed a procedure for solving multi-objective linear-programming problems. They have assumed that the decision maker has fuzzy goals for each objective function. Their procedure can obtain a non-dominated solution, which is also fuzzy-efficient. Konno and Ishii [12] have designed a preemptive OSSP with the fuzzy resource and allowable time. This problem has bi-criteria to be maximized, i.e., minimal satisfaction degree with respect to the processing intervals of jobs and minimal satisfaction degree of resource amounts used in the processing intervals. They have presented a solution procedure based on the network flow algorithm.

In scheduling problems, machines often should be prepared between jobs. This process is considered as a setup. The setup task of an operation can be separated from its corresponding

processing time. Therefore, it can be started in advance when the particular machine is free. Roshanaei et al. [24] have considered non-preemptive OSSPs with machine- and sequence-dependent setup times to minimize makespan. They have proposed two new advanced meta-heuristics, namely, multi-neighborhood SA search and hybrid SA, to solve this problem. Low and Yeh [16] have addressed an OSSP as a 0-1 integer programming model with the objective of minimizing the total job tardiness with some assumptions, such as independent setups and dependent removal times. They have proposed some hybrid genetic-based heuristics to solve the problem in an acceptable computing time. Mosheiov and Oron [20] have addressed batch scheduling problems on an *m*-machine open-shop with identical processing time jobs, machine- and sequence-independent setup time assumptions. The objectives were minimize makespan and flow time. They have proposed an O(*n*) time algorithm for the flow time minimization problem.

Allahverdi et al. [1] have surveyed the literature of setup time or cost in scheduling problems. They have classified scheduling problems into those with batching and non-batching considerations, and with sequence-independent and sequence-dependent setup times. Fazle Baki and Vickson [9] discussed two different pseudo polynomial dynamic programming recursions for each of the open shop and flow shop problems with one-operator, two-machine, and setup times for machines minimizing the weighted number of tardy jobs. Strusevich [27] has studied the problem of scheduling jobs in a two-machine open shop with group technology to minimize the makespan. A batch setup time on each machine is required before the first job is processed and when a machine switches from processing a job in some batch to a job of another batch. He has proposed a 5/4-approximation heuristic algorithm that creates a group technology schedule.

According to the literature review, there is little research using a fuzzy approach in OSSPs. Also, a combination of the fuzzy approach with multi-objective optimization in OSSP seems to receive less attention. So, in this paper, a bi-objective possibilistic mixed-integer linear programming (BOPMILP) model is designed for the OSSP and solved optimally. To solve medium to large sized problems, a novel multi-objective particle swarm optimization (MOPSO) algorithm is proposed to obtain a good approximate Pareto-optimal set.

The remainder of our work is organized as follows. The designed mathematical programming is presented in Section 2, followed by the proposed interactive fuzzy programming solution approach in Section 3. Section 4 elaborates the proposed MOPSO. Section 5 consists of the numerical examples, computational results and performance analysis. Finally, the conclusion is stated in Section 6.

2. Mathematical Formulation

Like all kinds of shop scheduling problems, OSSPs examined have are associated with *n* jobs to be processed on at most *m* machines. Two performance measures (i.e., weighted mean tardiness and weighted mean completion times) are considered in this problem and a set of feasible solutions are searched to optimize these measures.

2.1. Problem Assumptions

- Each job should be processed on at most *m* machines.
- At any time, at most one job can be processed on each machine.
- The overlap between operations of a job is not allowed.

- The processing order of operations is immaterial.
- The importance levels of the jobs are different and are not necessary to be less than 1.
- All jobs are available at time 0.
- No preemption is allowed. It means that the processing of a job on a machine cannot be interrupted.
- The machine breakdown is not permitted.
- Each job has its specified processing time and due date.
- Each operation is considered to have machine-dependent setup time and removal time. These times are assumed to be separated from their processing times.
- The setup task can be started in advance when a certain machine is free.
- All the processing times and due dates are considered to be fuzzy parameters with the triangular possibility distribution. A possibility distribution can be stated as the degree of occurrence of an event with imprecise data, and it is a common tool for modeling the ambiguous parameters.

2.2. Notations

N	Set of jobs to be processed; $N=\{1,2,\dots,n\}$; $\lvert N\rvert = n$
L	Set of machines; $L=\{1,2,\dots,m\}$; $\lvert L\rvert = m$
i,k	job indices; (i, k = 1,2,…,n)
j,h	machine indices; (j, h =1,2,…,m)
M	A large positive number
w_i	Tardiness penalty of job i
v_i	Priority of job i
O_{ij}	Operation of job i on machine j ; $\forall i \in N; \forall j \in L$
S_{ij}	Setup time of job i on machine j
$\tilde{p}_{ij}$	Fuzzy processing time of job i on machine j
R_{ij}	Removal time of job i on machine j
$\tilde{d}_i$	Fuzzy due date of job i
Ts_{ij}	Starting time of a setup task for operation O_{ij}
T_i	Tardiness of job i
C_i	Completion time of job i

$$Y_{ijh} = \begin{cases} 1, & \text{if } O_{ij} \text{ precedes } O_{ih} \text{ for job i} \\ 0, & \text{Otherwise.} \end{cases}$$

$$\forall i \in N; \forall j,h \in L, j \neq h$$

$$X_{ikj} = \begin{cases} 1, & \text{if } O_{ij} \text{ precedes } O_{kj} \text{ on machine j} \\ 0, & \text{Otherwise.} \end{cases}$$

$$\forall i,k \in N, i \neq k; \forall j,h \in L.$$

2.3. Mathematical Model

As mentioned in Section 2.1, all processing times and due dates are considered to be fuzzy parameters with the triangular possibility distribution as follows:

$$\tilde{p}_{ij} = \left(p_{ij}^{p}, p_{ij}^{m}, p_{ij}^{o}\right), \quad (1)$$

$$\tilde{d}_i = \left(d_i^{p}, d_i^{m}, d_i^{o}\right). \quad (2)$$

The decision maker can specify three parameters (p_{ij}^{p}, d_i^{p}), (p_{ij}^{m}, d_i^{m}) and (p_{ij}^{o}, d_i^{o}), which are most pessimistic values, most possible values and most optimistic values, respectively, as depicted in Figure 1.

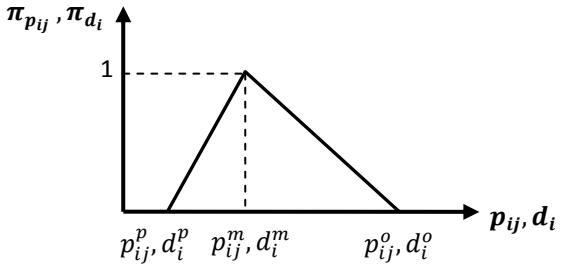

Figure 1. Triangular possibility distribution of fuzzy parameters $\tilde{p}_{ij}$, $\tilde{d}_i$

The bi-objective possibilistic mixed-integer linear programming (BOPMILP) problem is modeled by:

$$\text{Min } Z_1 = \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i} \tag{3}$$

$$\text{Min } Z_2 = \frac{\sum_{i=1}^{n} v_i C_i}{\sum_{i=1}^{n} v_i} \tag{4}$$

s.t.

$$Ts_{ij} + S_{ij} + \tilde{p}_{ij} + R_{ij} \le C_i \; ; \quad \forall i \in N; \forall j \in L \tag{5}$$

$$Ts_{ij} + S_{ij} + \tilde{p}_{ij} + R_{ij} - M(1 - Y_{ijh}) \le Ts_{ih} + S_{ih} \; ; \quad \forall i \in N; \; \forall j, h \in L, j \ne h \tag{6}$$

$$Ts_{ih} + S_{ih} + \tilde{p}_{ih} + R_{ih} - M \times Y_{ijh} \le Ts_{ij} + S_{ij} \; ; \quad \forall i \in N; \; \forall j, h \in L, j \ne h \tag{7}$$

$$Ts_{ij} + S_{ij} + \tilde{p}_{ij} + R_{ij} - M\left(1 - X_{ikj}\right) \le Ts_{kj} \; ; \quad \forall i, k \in N, i \ne k; \; \forall j \in L \tag{8}$$

$$Ts_{kj} + S_{kj} + \tilde{p}_{kj} + R_{kj} - M \times X_{ikj} \le Ts_{ij} \; ; \quad \forall i, k \in N, i \ne k; \; \forall j \in L \tag{9}$$

$$C_i - \tilde{d}_i \le T_i \; ; \forall i \in N \tag{10}$$

$$Y_{ijh} + Y_{ihj} = 1 \; ; \quad \forall i \in N; \; \forall j, h \in L \, , j \ne h \tag{11}$$

$$X_{ikj} + X_{kij} = 1 \; ; \quad \forall i, k \in N, i \ne k; \; \forall j \in L \tag{12}$$

$$Ts_{ij} \ge 0 \quad ; \; \forall i \in N; \forall j \in L \tag{13}$$

$$T_i, C_i \geq 0 \quad ; \ \forall i \in N \tag{14}$$

$$X_{ikj} Y_{ijh} \ \in \{0,1\} ; \quad \forall i, k \in N, i \neq k; \ \forall j, h \in L, j \neq h. \tag{15}$$

Two objective functions (i.e., weighted mean tardiness and weighted mean completion time) are shown by (3) and (4); $w_i$and v_i can be equal or not equal. Constraint (5) describes the completion time of job *i*. Constraints (6) and (7) express that operations O_{ij} and O_{ih}, of job *i* are not required to be consecutive, If operation O_{ij} is operation O_{ih}, then Y_{ijh} = 1; thus, the starting time for processing operation O_{ih} is greater than or equal to the completion time of operation O_{ij}. Constraints (8) and (9) characterize the constraint of the operational sequence of the operations which are processed on the same machine. Therefore, on machine *j*, if operation O_{ij} is processed before operation O_{kj}, then X_{ikj} = 1; the setup task of operation O_{kj} cannot be started until machine *j* has finished the removal task of operation O_{ij}. Constraint (10) describes the tardiness for each job *i* defined by:

$$T_i = max\left(0, C_i - \tilde{d}_i\right) \ ; \ \forall i \in N \tag{16}$$

Constraint (11) expresses the order of any two operations of a job; if $Y_{ijh} = 1$, then Y_{ihj} = 0; otherwise, $Y_{ijh} = 0$ and Y_{ihj} = 1. Constraint (12) expresses the order of any operation pairs (O_{ij},O_{kj}) on the same machine *j*; if $X_{ikj} = 1$, then X_{kij} = 0; otherwise , $X_{ikj} = 0$ and X_{kij} = 1. Constraint (13) implies that all jobs should be available for scheduling at time 0. Constraints (14) and (15) define the continuous and binary decision variables.

2.4. Equivalent Auxiliary Crisp Model

To convert fuzzy numbers in the left-hand sides of constrains (5)-(10) into crisp numbers, the weighted average method proposed by Lai and Hwang (1992a) is applied for the defuzzification process. The equivalent auxiliary crisp constraints are represented by:

$$Ts_{ij} + S_{ij} + w_1 p^p_{ij,\beta} + w_2 p^m_{ij,\beta} + w_3 p^o_{ij,\beta} + R_{ij} \leq C_i \ ; \forall i \in N; \forall j \in L \tag{17}$$

$$Ts_{ij} + S_{ij} + w_1 p^p_{ij,\beta} + w_2 p^m_{ij,\beta} + w_3 p^o_{ij,\beta} + R_{ij} - M(1 - Y_{ijh}) \leq Ts_{ih} + S_{ih}$$

$$\forall i \in N; \ \forall j, h \in L, j \neq h \tag{18}$$

$$Ts_{ih} + S_{ih} + w_1 p^p_{ih,\beta} + w_2 p^m_{ih,\beta} + w_3 p^o_{ih,\beta} + R_{ih} - M \times Y_{ijh} \leq Ts_{ij} + S_{ij}$$

$$\forall i \in N; \ \forall j, h \in L, j \neq h \tag{19}$$

$$Ts_{ij} + S_{ij} + w_1 p^p_{ij,\beta} + w_2 p^m_{ij,\beta} + w_3 p^o_{ij,\beta} + R_{ij} - M\left(1 - X_{ikj}\right) \leq Ts_{kj}$$

$$\forall i, k \in N, i \neq k; \ \forall j \in L \tag{20}$$

$$Ts_{kj} + S_{kj} + w_1 p^p_{kj,\beta} + w_2 p^m_{kj,\beta} + w_3 p^o_{kj,\beta} + R_{kj} - M \times X_{ikj} \leq Ts_{ij}$$

$$\forall i, k \in N, i \neq k; \ \forall j \in L \tag{21}$$

$$C_i - w_1 d_{i,\beta}^{p} + w_2 d_{i,\beta}^{m} + w_3 d_{i,\beta}^{o} \leq T_i; \quad \forall i \in N, \tag{22}$$

where, β is specified by the decision maker as the minimum acceptable possibility. Meanwhile, $w_1 + w_2 + w_3 = 1$, and w_1, w_2 and w_3 denote the weights of the most pessimistic, the most possible and the most optimistic value of fuzzy parameters, respectively. By using the most likely solution concept proposed by Lai and Hwang (1992b), these parameters can be set as to be: $w_1 = 1/6$, $w_2 = 4/6$, $w_3 = 1/6$. In addition, the following constraint is added to the model:

$$\beta \in [0,1]. \tag{23}$$

Thus, the auxiliary crisp bi-objective mixed-integer linear programming (BOMILP) model is formulated by:

$$\begin{aligned} &\min \ Z = [Z_1, Z_2] \\ &\text{s.t.} \\ &\quad \upsilon \in F_{(\upsilon),} \end{aligned} \tag{24}$$

Where, υ denotes a feasible solution vector consisting of all continuous and binary variables in the original model, and $F_{(\upsilon)}$ represents the feasible area involving crisp constraints (11) - (15) and (17) - (23).

3. Interactive Fuzzy Multi-Objective Decision Making Approach

Because of the difference between priority and tardiness penalty of each job, two objective functions considered in our study are conflicting. Thus, this problem cannot be solved by the use of single-objective optimization methods. There are three main groups of optimization methods based on the preference information specified by the decision maker for solving multi-objective decision making (MODM) problems:

- Priori optimization methods.
- Progressive optimization methods.
- Posteriori optimization methods.

Here, an efficient interactive fuzzy programming solution approach, which belongs to the first group, is used to obtain the Pareto-optimal solutions of resulting bi-objective crisp model. This method is called the TH method proposed by Torabi and Hassini [28]. The steps of the TH method are given below.

Step 1. For the fuzzy parameters, specify triangular possibiliy distributions and design the original BOPMILP model for the OSSP.

Step 2. Convert the fuzzy constraints into the corresponding crisp ones in order to achieve the auxiliary crisp BOMILP model, using β, the given minimum acceptable possibility level for fuzzy numbers.

Step 3. Solve the following MILP models in order to determine the positive ideal solution (PIS) and the negative ideal solution (NIS) for each optimization criterion:

$$Z_1^{PIS} = \text{Min}\ \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i} \qquad \text{s.t.} \quad \upsilon \in F_{(\upsilon)} \tag{25}$$

$$Z_1^{NIS} = \text{Max}\ \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i} \qquad \text{s.t.} \quad \upsilon \in F_{(\upsilon)} \tag{26}$$

$$Z_2^{PIS} = \text{Min}\ \frac{\sum_{i=1}^{n} v_i C_i}{\sum_{i=1}^{n} v_i} \qquad \text{s.t.} \quad \upsilon \in F_{(\upsilon)} \tag{27}$$

$$Z_2^{NIS} = \text{Max}\ \frac{\sum_{i=1}^{n} v_i C_i}{\sum_{i=1}^{n} v_i} \qquad \text{s.t.} \quad \upsilon \in F_{(\upsilon)}. \tag{28}$$

Obtaining the above ideal solutions requires solving four mixed-integer linear programs. To reduce the computing time, the negative ideal solutions can be determined by the use of the following heuristic rule:

$$Z_h^{NIS} = \text{Max}\ (Z_h(\upsilon_k^*))\ ; \quad h = 1,2\ , K=1, 2. \tag{29}$$

The results are shown in Table 1 as a *payoff* table.

Table1. Payoff table

	Z_1	Z_2
υ_1^*	Z_1^{PIS}	Z_2^{NIS}
υ_2^*	Z_1^{NIS}	Z_2^{PIS}

Step 4. Specify a linear membership function for each optimization criterion as follows:

It should be noted that the membership functions, $\mu_{Z_h}(\upsilon)$, denote the satisfaction degree of the h th optimization criterion for the feasible decision vector υ:

$$\mu_{Z_1}(\upsilon) = \begin{cases} 1, & Z_1 < Z_1^{PIS} \\ \frac{Z_1^{NIS} - Z_1}{Z_1^{NIS} - Z_1^{PIS}}, & Z_1^{PIS} \le Z_1 \le Z_1^{NIS} \\ 0, & Z_1 > Z_1^{NIS}. \end{cases} \tag{30}$$

$$\mu_{Z_2}(\upsilon) = \begin{cases} 1, & Z_2 < Z_2^{PIS} \\ \frac{Z_2^{NIS} - Z_2}{Z_2^{NIS} - Z_2^{PIS}}, & Z_2^{PIS} \le Z_2 \le Z_2^{NIS} \\ 0, & Z_2 > Z_2^{NIS}. \end{cases} \tag{31}$$

Figure 2 represents the graph of such membership functions.

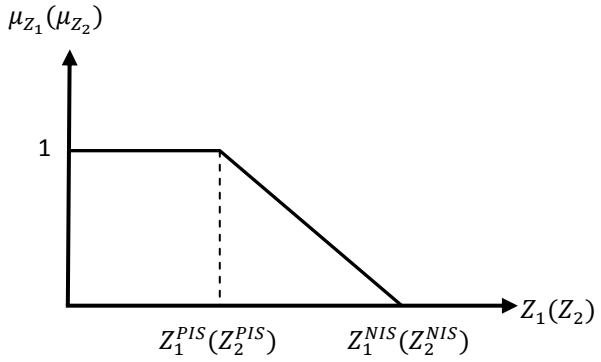

Figure 2. Linear membership function for $Z_1(Z_2)$

Step 5. Convert the auxiliary BOMILP model into an equivalent single-objective MILP Problem by the use of the following auxiliary crisp formulation:

$$\text{Max} \quad \lambda(\upsilon) = \gamma\lambda_0 + (1-\gamma)\sum_h \theta_h \mu_{Z_h}(\upsilon) \tag{32}$$

s.t.

$$\lambda_0 \leq \mu_{Z_h}(\upsilon) \ , \quad h = 1, 2 \tag{33}$$

$$\upsilon \in F_{(\upsilon)} \tag{34}$$

$$\lambda_0, \gamma \in [0,1]. \tag{35}$$

According to two optimization criteria of this problem, constraints (33) are written by:

$$Z_1^{NIS} - \frac{\sum_{i=1}^n w_i T_i}{\sum_{i=1}^n w_i} \geq \lambda_0 \left(Z_1^{NIS} - Z_1^{PIS}\right), \tag{36}$$

$$Z_2^{NIS} - \frac{\sum_{i=1}^n v_i C_i}{\sum_{i=1}^n v_i} \geq \lambda_0 \left(Z_2^{NIS} - Z_2^{PIS}\right), \tag{37}$$

Where, $\mu_{Z_h}(X)$ is the satisfaction degree of the hth objective function and $\lambda_0 = \min_h \{\mu_{Z_h}(X)\}$ is the minimum satisfaction degree of the objectives. In addition, θ_h denotes the importance level of the hth objective function such that $\sum_h \theta_h = 1$, $\theta_h > 0$. The θ_h parameters are determined linguistically by the decision maker based on her/his preference. Moreover, γ is the coefficient of compensation. By changing the value of this parameter in the interval [0,1], the TH method can obtain both unbalanced and balanced compromised solutions. It means that for higher values of γ, the solution method results bigger lower bounds for the satisfaction degrees of objectives (λ_0) for a given sample example. These solutions are balanced compromised solutions. These kinds of solutions can be more appropriate when the importance levels of all objective functions are equal. On the other hand, for lower values of γ, the solution method results in solutions with bigger satisfaction degrees for some objectives with higher importance

levels than others. These solutions are unbalanced compromised solutions. These kinds of solutions can be more appropriate when the importance levels of the objective functions are different.

Step 6. Apply the given coefficients (θ_h ,γ). Solve the equivalent single-objective MILP problem. If the decision maker is satisfied with the obtained efficient compromised solution, then stop. Otherwise, change the value of some controllable parameters (β and γ), and go back to Step 2.

The validity and efficiency of the proposed OSSP model is checked and shown by the use of several numerical instances in Section 5. These instances are generated randomly by the use of a classic approach in the literature.

It should be mentioned that changing the values of controllable parameters of equivalent single-objective MILP problem (i.e., (32)-(37)) results in obtaining a set of non-dominated solutions as Pareto-optimal solutions for the BOMILP model given by (24). It means that each optimal solution of the final MILP model is an efficient solution to the BOMILP model.

4. The Proposed MOPSO

As mentioned in Section 1, the OSSP studied have belongs to a class of NP-hard problems. Thus, to solve medium to large-sized problems, an efficient multi-objective particle swarm optimization (MOPSO) algorithm is proposed.

4.1. Classic PSO

Particle swarm optimization (PSO) has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life), in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related, however, to evolutionary computing, and has ties to both GA and evolutionary programming [11].

A swarm is composed of particles such as birds, fishes, bees, etc. Each particle searches the area for food with its velocity and always remembers the best position found. This value is called *pbest*. In addition, each member of the swarm knows the best position found by its best informant or by the group globally. This value is called *gbest*. Therefore, there are three fundamental elements for the calculation of the next displacement of a particle: 1) according to its own velocity, 2) towards its best performance, and 3) the best performance of its best informant. The way in which these three vectors are combined linearly via confidence coefficients is the basis of all versions of the "classic" PSO [6].

4.2. Advantages of PSO

- The main advantages of PSO are its simplicity (both conceptually, and at the implementation level), its ease of use and its high convergence rate. In fact, PSO is a good candidate to design an "ultra-efficient" multi-objective evolutionary algorithm [7].
- PSO is an evolutionary algorithm, and its population to the population approach enables the search to escape from the local optima.
- The inertia weight in the equations of motion controls the exploration and exploitation. A larger w can prevent particles from becoming trapped in local optima, and a smaller w encourages particles exploiting the same search area [26].

4.3. The Proposed MOPSO (PMOPSO)

To solve the BOMILP model presented in Section 2, an efficient MOPSO is designed as follows.

4.3.1. Solution Representation

Two kinds of a solution representation are used in this algorithm: (1) Operation-based array or permutation list, and (2) Continuous representation. Each particle has these two representations simultaneously, each of which is used in different steps of the proposed MOPSO (PMOPSO).

4.3.1.1. Operation-Based Representation

This kind of representation is typically applied in the literature of OSSP. The operation-based array or permutation list is a single-row array. Each job should be processed on each machine once and the processing order of operations is immaterial (i.e., a feasible sequence) for a problem with n jobs and m machines. It is a single-row array consisting of $n \times m$ elements that is equal to the number of the operations. In this representation, operations are listed in the relative order by which they are scheduled. Figure 3 illustrates a permutation list. In this figure, p shows the position of an operation in the list.

p=1	p=2	p=3	p=4	...	p= $n \times m$
O_{43}	O_{16}	O_{22}	O_{13}	...	O_{ij}

Figure 3. Operation-based representation/permutation list

The common decoding procedure using a permutation list is as follows. Each operation, according to its corresponding relative order, is scheduled at the earliest time the job and the machine are both available. For instance, in order to decode the permutation list of Figure 3, job 4 is first scheduled on machine 3. Then, job 1 is scheduled on machine 6 and so on. It is supposed that $\max\{C_{ij}\}, \forall i$, is equal to the maximum completion time of jobs on machine j and $\max\{C_{ij}\}, \forall j$, is equal to the maximum completion time of job i. Thus, the starting time and completion time of each operation is calculated by:

$$Ts_{ij} = \max\ \{0\ ,\ \max_{\forall i}\{C_{ij}\}\ ,\ \max_{\forall j}\{C_{ij}\}\ \}\ , \tag{38}$$

$$C_{ij} = Ts_{ij} + S_{ij} + p_{ij} + R_{ij}. \tag{39}$$

By the use of common decoding procedure, a semi-active schedule is obtained. The set of semi-active schedules is very large and has poor quality in terms of the objective functions considered in this study. Therefore, designing an effective decoding scheme to decode the particle position into a schedule within a smaller schedule set with a better solution quality will improve the efficiency of PMOPSO algorithm. So, a modified decoding procedure by an operation-based array is proposed. This procedure can achieve an active schedule or a semi-active schedule with a higher quality in terms of the two objective functions. In fact, this heuristic procedure reduces the search area in the solution space but does not exclude the

optimal solution. Thus, the execution time of the proposed MOPSO algorithm is spent on exploiting the solution space appropriately and searching the efficient solutions in a set of high quality schedules. The steps of the proposed modified decoding scheme are presented below.

Two notations should be first introduced before presenting the steps of the procedure.

TJA_i Earliest time job *i* is available for the next operation

TMA_j Earliest time machine *j* is available/free again

Step 1. Consider a permutation list as shown in Figure 3. At first, TJA_i and TMA_j , $\forall i, \forall j$, are equal to 0.

Step 2. Select the operation that is in the first position (p=1) and schedule it by the use of Equations (38) and (39). Select the operations of different jobs and different machines such as O_{ij}, *and* O_{kh}, according to the corresponding relative order of the operations. Schedule these operations by the use of Equations (38) and (39).

Step 3. Update the TJA_i and TMA_j , $\forall i, \forall j$, such that

$TJA_i = \text{Max}\{C_{ij}\}$, $\forall j$,
$TMA_j = \text{Max}\{C_{ij}\}$, $\forall i$.
Check the array. If all of the operations are scheduled then stop.

Step 4. According to the corresponding relative order of the operations, consider an operation such as O_{ij} that is not scheduled yet. If $TMA_j < TJA_i$ then go to Step 5; otherwise, schedule this operation by the use of Equations (38) and (39), and go back to Step 3.

Step 5. Set $TMA_j = TMA_j + S_{ij}$ and delete S_{ij} from Equation (39), and schedule this operation by the use of Equations (38) and (39), and then go back to Step 3.

The following instance can show the performance of this procedure. Consider a problem with three jobs and three machines. Each job must be processed on every machine once. Thus, there are nine operations to be scheduled. Assume that the permutation list illustrated in Figure 4 depicts a feasible sequence of this example. Figure 5 shows the Gantt chart of a feasible solution obtained by the use of the common decoding procedure. Figure 6 shows the Gantt chart of a feasible solution obtained by the use of the proposed decoding procedure. The dark color represents the setup task and the grey color represents the removal task. The results that indicate the schedule obtained by the use of the common procedure has poor quality in terms of two performance measures but the schedule obtained by the use of the proposed procedure is an active schedule and has a higher quality.

It is assumed:

d_1= 10 , d_2= 7 , d_3= 12
w_1= 1 , w_2= 3 , w_3= 2
v_1= 3 , v_2= 2 , v_3= 1.
So, the objective functions values are:

In Figure 5, we have:

$$Z_1 = \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i} = 9.83, \qquad Z_2 = \frac{\sum_{i=1}^{n} v_i C_i}{\sum_{i=1}^{n} v_i} = 15.5.$$

In Figure 6, we have:

$$Z_1 = \frac{\sum_{i=1}^{n} w_i T_i}{\sum_{i=1}^{n} w_i} = 2.33, \qquad Z_2 = \frac{\sum_{i=1}^{n} v_i C_i}{\sum_{i=1}^{n} v_i} = 9.83.$$

O_{23}	O_{13}	O_{11}	O_{21}	O_{33}	O_{31}	O_{12}	O_{32}	O_{22}

Figure 4. Operation-based array for the example with n = 3 and m = 3

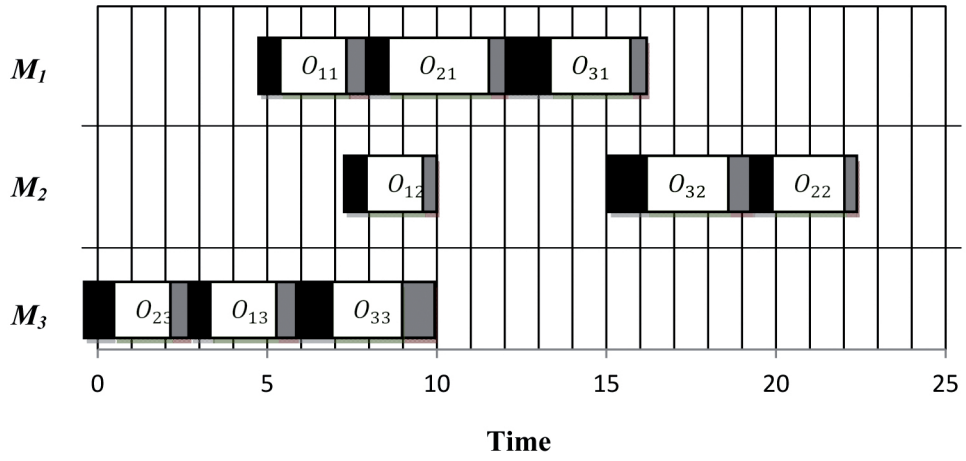

Figure 5. Gantt chart of the feasible schedule obtained by the use of the common decoding procedure

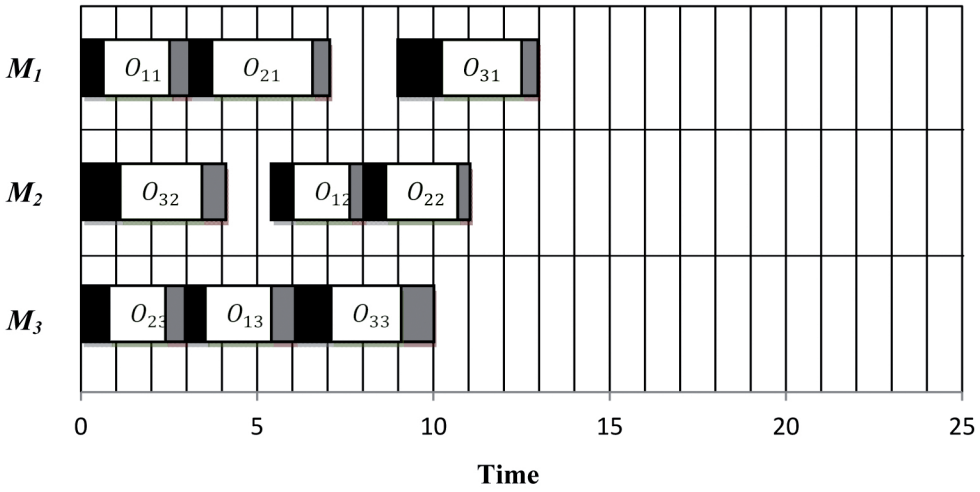

Figure 6. Gantt chart of the feasible active schedule obtained by the use of the proposed decoding procedure

4.3.1.2. Continuous Representation

Continuous representation is a single-row array including $n \times m$ real numbers, when there are n jobs and m machines. Each of these real values indicates a particle's position in each dimension of the $n \times m$-dimensional space that they move in. For these values, an interval $[0, X_{max}]$ puts a limit on the minimum and maximum distances each particle is permitted to move in each dimension. In our work here, the continuous representation matrix for particle k, $[X]k_{1\times nm}$, is composed of $n \times m$ random continuous numbers between $[0, X_{max}]= [0, 4]$.

As mentioned before, each particle has two representations simultaneously. The position of a particle does not represent a solution for the original problem. This type encoding scheme is applied during the execution of the main loop of PSO and should be transformed into an operation-based array in order to obtain a solution of the OSSP. Meanwhile, a permutation list needs to be transformed into a continuous representation used in PSO. To convert the operation-based array into the continuous representation, at first, select $n \times m$ random numbers between 0 and 4. Then, sort these numbers in ascending order. Assign the smallest number to $p = 1$, assign the next smallest to $p = 2$ and so on [23]. Table 2 and Figure 7 show this transformation for the array given in Figure 4. In addition, to construct a permutation list, the operation standing on the first position (i.e., p=1) is the operation equivalent to the smallest number, the operation stand on the second position is the operation equivalent to the next smallest number of the continuous representation, and so on.

Table 2. A sample set of random numbers

No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9
1.65	3.8	2.44	2.9	1.18	2.3	0.24	3.24	3.52

0.24	1.18	1.65	2.3	2.44	2.9	3.24	3.52	3.8

Figure 7. Continuous representation of Figure 4

4.3.2. Initialization

The initial position of a particle can be generated by using some common ways (i.e., random generation, local search methods, meta-heuristic algorithms, etc.). In this paper, a simple local search is applied to generate the initial set of solutions or the population. The algorithm goes as follows. Construct a set of random sequences of operations as permutation lists (as many as the number of the particles). Generate all the feasible sequences in the neighborhood of each permutation list by using the swapping procedure of two adjacent or non-adjacent operations as a movement function. The best feasible solution obtained for each particle using these neighborhood searches is a member of the initial population or initial particle positions.

The best solution found by solving MODM problems is an efficient solution of the Pareto-optimal frontier. These Pareto-optimal solutions improve all the optimization criteria

simultaneously. They dominate other solutions in the feasible region. Consider the following multi-objective model:

$$\min \ F(x) = \{f_1(x), f_2(x), \dots, f_n(x)\} \tag{40}$$

s.t.

$$g_k(x) \begin{bmatrix} \leq \\ \geq \\ = \end{bmatrix} 0, k = 1,2, \dots, m \tag{41}$$

$$x \in E^n. \tag{42}$$

A Solution x is said to dominate a solution $\hat{x}$ if and only if:

$$f_i(x) \leq f_i(\hat{x}) \ , \forall i, \tag{43}$$

$$f_j(x) < f_j(\hat{x}) \ . \ \exists j \tag{44}$$

By using transformation procedure to build continuous representation described in the previous section, each member of the generated population is prepared to be applied in the main loop of PSO as the particles, initial positions.

4.3.3. Generating Initial Velocities

In PSO, each particle requires the initial velocity equivalent to its initial position in order to make a movement in a search space. Since the position matrix of the particles is a single row array made of real values, the continuous values are assigned to velocities as well. In this paper, the initial velocity matrix is a single-row array made of $n \times m$ continuous values between v_{min} and v_{max} , which are set to -4 and 4, respectively. Each value is computed by

$$v_{ij}(k) = v_{min} + (v_{max} - v_{min})^* r_1 , \tag{45}$$

Where, r_1 is a random variable between 0 and 1. The range of acceptable velocity values is chosen such that $|v_{ij}(k)| \leq X_{max}$, where $v_{ij}(k)$ is the velocity of job i on machine j for particle k.

4.3.4. Updating Velocity and Position Values

For solving multi-objective problems by PSO, the most important item that should be paid more attention is the selection procedure of the *gbest* and *pbest* for each particle when the velocity and position of particles should be updated.

The analogy of PSO with evolutionary algorithms makes it evident that using a Pareto ranking scheme can be the straightforward way to extend the approach to handle multi-objective optimization problems. The historical record of best solutions found by a particle (i.e., an individual) can be used to store non-dominated solutions generated in the past, this being similar to the notion of elitism used in evolutionary multi-objective optimization. The use of global attraction mechanisms combined with a historical archive of previously found non-dominated vectors motivate convergence towards *Ptrue* [7].

For each particle, there is a *pbest* archive that can save all the best positions found by the particle. Actually, these positions are non-dominated solutions and each particle has its own

approximate Pareto solutions archive. When a new solution is obtained, it is compared with the ones in *pbest* archive based on the domination principle. If this new solution dominates any other in the *pbest* archive, that archived solution is removed from the *pbest* archive and the new solution is placed in the archive. Moreover, there is a *gbest* archive for the group and the global best solutions are saved. This set of Pareto solutions are obtained using the fast non-dominated sorting procedure proposed by Deb et al. [8]. Updating the *gbest* archive is done like updating the *pbest* archive. That is, if a new solution dominates any other in the *gbest* archive, that archived solution is removed from the *gbest* archive and the new solution is placed in the archive.

Here, to select the *gbest* and *pbest* of each particle for updating velocity and position values, a new procedure from the literature is applied [3]. To maintain diversity in the search and escape from the local optima, each *gbest* has a chance to be selected. The steps of this selection procedure are presented below:

Step 1. Divide the number of the particles into the number of the *gbest archive*.

Step 2. Consider the remainder. If the remainder is equal to 0, then the number of times for each member of *gbest* can be selected to update the velocity of the particles equal the sub-multiple value, and go to Step 4; otherwise, go to Step 3.

Step 3. Calculate the crowding distance proposed by Deb et al. [8] for each member of *gbest*. Sort these obtained values in descending order and choose the first *k* members of *gbest* (as many as the remainder value). The number of times each member of *gbest* except these *k* members of *gbest* can be selected to update the velocity of the particles is equal to the sub-multiple value. However, these *k* members of *gbest* have one more chance than other members to be selected. G Then, go to Step 4.

Step 4. Select the *gbest* and a *pbest* for each particle. To assign the *gbest* to a particle, the minimum Euclidean distance of that particle from the *gbests*, which still has a chance to be selected, is considered. It means that the nearest *gbest* to each particle, which still has a chance, is chosen. To assign a *pbest* to a particle, the *pbest* with the maximum Euclidean distance from the *gbest* assigned to that particle is chosen. Stop.

The following numerical example can describe the mentioned steps clearly. Assume that a PSO algorithm has 90 particles and the number of members of *gbest* in the *gbest* archive is equal to 7. If we divide 90 into 7, the sub-multiple value is equal to 12. So, each member of *gbest* can be selected 12 times. However, the remainder does not equal 0. It means that 6 particles do not have *gbest* values for updating velocity and position values. In this case, we calculate the crowding distance for each member of *gbest* and sort the resulting numbers in descending order. The first 6 members of *gbest* with large crowding distance values are considered. These members can be selected 13 times.

Finally, the velocity and position values of each particle with the corresponding selected *pbest* and *gbest* are updated by the use of the equations of velocity and motion computed by

$$v_{id}(t+1)= w(t)v_{id}(t)+c_1r_1[p_{id}(t)-x_{id}(t)]+c_2r_2[p_{gd}(t)-x_{id}(t)] \tag{46}$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1),$$

Where, $w(t)$ is the inertia weight representing the particle's preference to continue moving in the same direction it was going in the previous iteration. This value is updated as follows: $w(t)=w(t\text{-}1)\times\alpha$, where α is a decrement factor. In addition, c_1r_1 and c_2r_2 are respectively confidence coefficients in a particle best performance and that of its best informant, t represents the iteration number, r_1, r_2 are the random variables in [0,1], i is the index of the particle and d is the index for the search space dimensions.

When the PMOPSO is iterated as many as a pre-specified value, the multi-objective optimization process is terminated and the set of solutions of the final *gbest* archive is reported as the Pareto/efficient solutions of the BOMILP problem.

5. Computational Results

To examine and analyze the validity and efficiency of the mathematical model presented in Section 2, the proposed MODM method discussed in Section 3, and the performance of the PMOPSO given in Section 4, several small, medium and large size numerical instances are generated randomly using a classic approach existing in the literature. Small size problems are solved exactly by using the Lingo software in a few minutes and the results obtained by the TH method are analyzed. Since for more than 4 jobs and 4 machines Lingo cannot yield an optimal solution, even after several hours running, in order to solve medium to large size problems, PMOPSO is used. Finally, the performance of this method is compared with the common MOPSO (CMOPSO) by the use of the design of experiments (DOEs) based on three comparison metrics. These two algorithms are coded in Turbo C++ 4.5.

5.1. Generating Numerical Examples

The processing times and due dates are considered as symmetric triangular possibilistic distributions calculated by:

$$\tilde{P}_{ij} = \left(P_{ij} - u_{ij}, P_{ij}, P_{ij} + u_{ij}\right)$$
$$\tilde{d}_i = (d_i - u_i, d_i, d_i + u_i)\ ,$$

where P_{ij} , d_i are the most possible values and u_{ij}, u_i represent the extension values of these fuzzy numbers. Here, we use, a classic approach from the literature, proposed by Loukil et al. [17], to generate the numerical instances for the scheduling problems: The most possible values of processing times and due dates are generated using the uniform distribution of [0,100] and $\left[P\left(1-T-\frac{R}{2}\right), P\left(1-T+\frac{R}{2}\right)\right]$, respectively, where $P=(m+n-1)\bar{P}$, with $\bar{P}=\sum_{i=1}^{n}\sum_{j=1}^{m}p_{ij}/(n\times m)$ as the mean of most possible values of the processing times. Two parameters R and T take their values in the set of {0.2, 0.6, 1} and {0.4, 0.6, 0.8}, respectively. The values of u_{ij} and u_i are uniformly distributed in the interval [2, 10]. Also, the setup times and removal times are random variables between 15 and 55. Finally, the values of priority and tardiness penalty, v_i and w_i, are selected randomly between 1 and 9.

By the use of the above-mentioned method, to generate small size examples, the number of jobs can be 4 or 5 or 6 and the number of machines can be equal to 2 or 3 or 4. Also, to generate medium to large size examples, the number of jobs can be equal to 10 or 12 or14 or 20 or 30

and the number of machines can be any value between 5 to 9. For each of these random instances, the open shop environment is defined as numbers of jobs×machines. For instance, the problem 6×2 indicates an open shop environment with 6 jobs and 2 machines.

5.2. Parameter Settings and Assumptions

Several experiments are implemented in different sizes and different sets of controllable parameters. From which, the effective values in terms of solution quality and computational time are chosen. The parameters of the TH method and those of the generating instances approach are as follows. The final MILP model obtained in Section 3 has very little sensitivity to the parameter β. Thus, the values of this parameter are set arbitrarily to 0.4 and 0.8. Parameter γ takes its values in the set {0, 0.1, …, 1}. Also, it is assumed that the preference information corresponding to the relative importance of the objective functions are specified linguistically by the decision maker as $\theta_1 = \theta_2$ and $\theta_1 > \theta_2$. So, the values of these parameters in the first case are $\theta_1 = \theta_2 = 0.5$, and in the second case are $\theta_1 = 0.7$, $\theta_2 = 0.3$. The values of the controllable parameters R and T are specified later.

According to the values proposed in the literature [3, 6, 11], the parameters of the PMOPSO algorithm (i.e., c_1 and c_2, $w(t)$, the reducing factor (α), the number of particles/solutions in each iteration (N), and the number of iterations; (i.e., stopping criteria) are set to $c_1 = c_2 = 2$, $w(t) = 0.975$, $\alpha = 0.9$, $N = 100$, and the number of iterations is set to 50. Furthermore, each example is solved 10 times independently.

5.3. Computational Results for Small Size Problems

Different combinations of jobs and machines in small sizes are considered to make three types of sample examples. For each kind of these examples, four numerical instances are generated randomly. The characteristic of sample examples and the values of controllable parameters for each numerical instance are presented in tables 3 and 4.

The values of positive ideal solutions (Z_i^{PIS}) and negative ideal solutions (Z_i^{NIS}) and the computing times (in seconds) for each numerical instance are illustrated in Table 6. It should be noted that to solve the final MILP model by the use of Lingo 8, the values of PIS and NIS of each objective function is calculated as described in Section 3. $\bar{T}_w$ and $\bar{C}_v$ represent weighted mean tardiness and weighted mean completion times, respectively. The final MILP model are exactly solved for each numerical example by the Lingo 8 software package, applying the obtained values of PIS and NIS corresponding to each optimization criterion. The results of a given example (i.e., 4×4-d) are presented in Table 7.

Table 3. Characteristic of small size sample examples

Sample example	Representation	Number of jobs	No. of machines	No. of decision variables	No. of constraints
1	6×2	6	2	120	235
2	5×3	5	3	145	291
3	4×4	4	4	152	309

Table 4. Values of controllable parameters for each numerical instance

Numerical instances	R	T	β
a	0.2	0.8	0.4
b	0.6	0.6	0.4
c	1	0.4	0.8
d	0.2	0.4	0.8

Considering the computational results for small size problems, the following observations are made.

- The TH method performs correctly and efficiently in more cases with different relative importance of optimization criteria. It means that, according to the relative importance, the solutions found by this method are unbalanced compromised solutions. However, in some cases, the TH method does not perform well and the satisfaction degree of the objective function with a lower importance is more than that of the objective function with a higher importance.
- When the relative importance is the same, in some cases, the satisfaction degree of the objective functions are very close to each other. It means the that solutions found by the TH method are approximately balanced compromised solutions.
- As mentioned in Section 3, each optimal solution of the final MILP model is an efficient solution to the BOMILP model. Thus, by changing the values of controllable parameters (i.e., γ and θ) in more cases, two or three Pareto-optimal solutions are found by the TH method. Figure 8 depicts the Pareto-optimal frontier of the example 4×4-d.

5.4. Computational Results for Medium to Large Size Problems

In this section, the performance of the PMOPSO algorithm with the characteristics and features described in Section 4 is compared with the CMOPSO algorithm, not having these features. Therefore, the differences between the algorithms are as follows:

- For CMOPSO, the initial population is generated randomly.
- The common decoding procedure by permutation list is used in CMOPSO.
- In CMOPSO, the *gbest* with the maximum crowding distance is selected for updating the velocity of particles.
- The *pbest* for updating the velocity of particles are selected randomly in CMOPSO.

It should be noted that for both PMOPSO and CMOPSO, the fuzzy parameters of OSSPs are converted into the crisp parameters by use of the approach discussed in Section 2. Different combinations of jobs and machines in medium and large sizes are considered to generate five examples randomly. These examples are solved by the PMOPSO and the CMOPSO algorithms. The characteristic of the sample examples and the values of the controllable parameters are presented in Table 5. By solving the above sample examples, the performance of the PMOPSO is compared with CMOPSO by the use of the design of experiments (DOEs) based on three comparison metrics.

Table 5. Characteristic and values of controllable parameters of medium to large size example

Sample example	representation	Number of jobs	Number of machines
1	10×5	10	5
2	12×6	12	6
3	14×7	14	7
4	20×8	20	8
5	30×9	30	9
$R = 0.2$, $T = 0.4$, $\beta = 0.8$			

5.4.1. Performance Evaluation Metrics

To evaluate the performance of the proposed MOPSO (i.e., PMOPSO) and common MOPSO (i.e., CMOPSO), three useful comparison metrics in the literature are taken into account. These metrics are described below:

- *Quality Metric* (*QM*): This metric represents the number of Pareto-optimal solutions found by each algorithm. Based on this metric, the algorithm that finds more Pareto-optimal solutions has a higher quality. However, some Pareto solutions of an algorithm may dominate those obtained by other algorithms. So, the number of final non-dominated solutions found by each algorithm is noted.
- *Diversity Metric* (*DM*): This metric is applied to estimate the spread of the obtained Pareto-optimal solution set found by each algorithm and is calculated by

$$D = \sqrt{\sum_{i=1}^{n} \max(|x_i - y_i|, \vec{x}, \vec{y} \in F)}, \tag{47}$$

  where, F denotes the set of obtained Pareto-optimal solutions and n is the dimension of the solution space that is equal to the number of optimization criteria.
- *Spacing Metric* (*SM*): To estimate the uniformity of the spread of the points of the obtained Pareto-optimal solution frontier, the spacing metric is applied and calculated by

$$S = \frac{\sum_{i=1}^{N-1} |d_i - \bar{d}|}{(N-1)\bar{d}}, \tag{48}$$

Where, d_i denotes the Euclidean distance between the consecutive solutions of the obtained the Pareto-optimal solution set, $\bar{d}$ represents the mean value of all Euclidean distances and N is the number of obtained Pareto-optimal solutions.

The computational results corresponding to these three performance metrics for medium to large size problems are presented in tables 8-10.

5.4.2. Design of Experiments

To analyze the results of solving medium to large size problems, two-factor factorial experiments are designed to examine the effect of the two solution methods on five test problems with respect to each of the comparison metrics, with 10 times executions. Thus, the following statistical linear model [19] can represent the results of tables 8-10:

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \varepsilon_{ijk} \begin{cases} i = 1,2 \\ j = 1,2,\dots,5 \\ k = 1,2,\dots,10 \end{cases} \quad (49)$$

where, μ is a common effect for the whole experiment, τ_i is the effect of the i th solution method, β_j is the effect of the j th test problem, $(\tau\beta)_{ij}$ is the interaction of the i th solution method and the j th test problem and ε_{ijk} is the random error.

It should be mentioned that the adequacy of factorial designs is checked first. The hypothesis test is considered as follows. The row treatment (i.e., solution methods) effects are equal to 0. The column treatment (i.e., test problems) effects are equal to 0. The interactions are equal to 0. In addition, the significance level (α) is set to be 0.05. Thus, we set the following tests:

$$H_0: \tau_1 = \tau_2 = 0 \quad (50)$$
$$H_1: \tau_1 \neq \tau_2$$

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_5 = 0 \quad (51)$$
$$H_1: \text{at least one } \beta_j \neq 0$$

$$H_0: (\tau\beta)_{ij} = 0 \quad (52)$$
$$H_1: \text{at least one } (\tau\beta)_{ij} \neq 0$$

The ANOVA test is performed by the use of the SAS 9.1 software. Table 11, gives the ANOVA results for the QM. Considering the results, the P-Value for the solution method main effect is less than α=0.05. So, the solution method effect is significant. It means there is a significant difference between the mean values for the two solution methods. However, the test problem effect and the interaction are not significant. Figure 9 indicates that more Pareto solutions are found by PMOPSO, and this method outperforms CMOPSO.

Table 12 shows the ANOVA result outperforms for the DM. Considering the results, the *P*-Value for the solution method main effect and that for the test problem main effect is less than α=0.05. So, the solution method effect and test problem effect are significant. In addition, the test problem effect for diversity is more than the effect of the solution method. However, the interaction is not significant. The diversity of solutions found by PMOPSO is more than CMOPSO. Thus, PMOPSO performs better than CMOPSO as shown in Figure 10.

The ANOVA result for the SM, which are illustrated in Table 13, shows that the *P*-Value for the solution method main effect and that for the test problem main effect is less than α=0.05. Therefore, the effects of the solution method and test problem are significant. Furthermore, the solution method effect for spacing metric is more than the test problem effect. Moreover, the interaction is not significant. The spacing metric value for solutions found by the PMOPSO is less than CMOPSO as shown in Figure 11. So, the performance of PMOPSO is better than COMPSO.

Considering the above observations, all the three test of hypotheses theses show the significant effect and better performance of PMOPSO over CMOPSO based on all comparison metrics.

6. Conclusion

A, bi-objective possibilistic programming formulation was presented for an open shop scheduling problem. An interactive approach proposed by Torabi and Hassini (i.e., the TH method) was applied to transform the original model into an auxiliary single-objective one and achieve the Pareto-optimal solutions for small size problems. Examining the results, in more cases, the TH method performed well according to the relative importance of the objective functions, finding two or three Pareto-optimal solutions. To find a good approximate Pareto-optimal set, an efficient multi-objective particle swarm optimization algorithm (PMOPSO) was proposed. This algorithm exploited new selection regimes of the literature for the global best (i.e., *gbest*) and personal best (i.e., *pbest*). Moreover, a modified decoding scheme was designed in order to reduce the search area in the solution space and a local search was proposed to generate the initial population. The medium to large size instances were solved by the use of the PMOPSO algorithm and the obtained results were compared with the results of the CMOPSO not having the features of the PMOPSO. The computational results were analyzed by the use of the design of experiments based on three comparison metrics (i.e., QM, DM and SM) as response variables. The results of ANOVA showed the significant effect and a better performance of PMOPSO over than the CMOPSO.

References

[1] Allahverdi, A., Ng, C.T., Cheng, T.C.E. and Kovalyov, M.Y. (2008), A survey of scheduling problems with setup times or costs, *European Journal of Operational Research*, 187, 985-1032.

[2] Andersen, M., Bräsel, H., Mörig, M., Tusch, J., Werner, F. and Willenius, P. (2008), Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop, *Mathematical and Computer Modeling*, 48, 1279-1293.

[3] Aramoon-Bajestani, M., Torabi, S.A. and Tavakkoli-Moghddam, R. (2009), A particle Swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling with a secondary resource constraint, Submitted to Information Sciences.

[4] Blum, C. (2005), Beam-ACO hybridizing ant colony optimization with beam search: an application to open shop scheduling, *Computers & Operations Research*, 32, 1565-1591.

[5] Blum, C. and Sampels, M. (2004), An ant colony optimization algorithm for shop scheduling problems, *Journal of Mathematical Modeling and Algorithms*, 3, 285-308.

[6] Clerc, M. (2006), Particles Swarm Optimization, ISTE Ltd.

[7] Coello Coello, C.A., Lamont, G.B. and Van Veldhuizen, D.A. (2007), Evolutionary algorithms for solving multi-objective problems, 2nd edition, Springer Science+Business Media, LLC.

[8] Deb, K., Pratap A., Agarwal, S. and Meyarivan, T. (2002), A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6, 182-197.

[9] Fazle Baki, M. and Vickson, R.G. (2004), One-operator, two-machine open shop and flow shop problems with setup times for machines and weighted number of tardy jobs objective, *Optimization Methods and Software*, 19, 165-178.

[10] Jimenez, M. and Bilbao, A. (2008), Pareto-optimal solution in fuzzy multi-objective linear programming, *Fuzzy Set and System*, 160, 2714-2721.

[11] Kennedy, J. and Eberhart, R. (1995), Particle swarm optimization, *The 1995 IEEE international conference on neural networks,* 4**,** 1942-1948.

[12] Konno, T. and Ishii, H. (2000) , An open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint, *Fuzzy Sets and Systems*,109, 141-147.

[13] Lai, Y.J., Hwang, C.L. (1992a), Fuzzy Mathematical Programming: Methods and Applications, Springer-Verlag, Berlin.

[14] Lai, Y.J. and Hwang, C.L. (1992b), A new approach to some possibilistic linear programming problem, *Fuzzy Sets and Systems*, 49, 121-133.

[15] Liaw, C.F. (2003), An efficient tabu search approach for the two-machine preemptive open shop scheduling problem, *Computers & Operations Research*, 30, 2081-2095.

[16] Low, C. and Yeh, Y. (2008), Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated, *Robotics and Computer-Integrated Manufacturing*, 25, 314-322.

[17] Loukil, T., Teghem, J. and Tuyttens, D. (2005), Solving multi-objective production scheduling problems using metaheuristics, *European Journal of Operational Research*, 161, 42-61.

[18] Matta, M.E. (2009), A genetic algorithm for the proportionate multiprocessor open shop, *Computers & Operations Research*, 36, 2601-2618.

[19] Montgomery, D.C. (2001), Design and Analysis of Experiments, 5th edition, John Wiley & Sons, INC, New York.

[20] Mosheiov, G. and Oron, D. (2008), Open-shop batch scheduling with identical jobs, *European Journal of Operational Research*, 187, 1282-1292.

[21] Prins, C. (2000), Competitive genetic algorithms for the open-shop scheduling problem, *Mathematical Methods of Operations Research*, 52, 389-411.

[22] Puente, J., Díez, H.R., Varela, R., Vela, C.R. and Hidalgo, L.P. (2004), Heuristic rules and genetic algorithms for open shop scheduling problem, *Lecture Notes in Artificial Intelligence (LNAI)*, 3040, 394-403.

[23] Rahimi-Vahed, A.R. and Mirghorbani, S.M. (2007), A multi-objective particle swarm for a flow shop scheduling problem, *Journal of Combinatorial Optimization*, 13, 79-102.

[24] Roshanaei, V., Seyyed Esfehani, M. M. and Zandieh, M. (2009), Integrating non-preemptive open shops scheduling with sequence-dependent setup times using advanced metaheuristic, *Expert Systems with Applications*, 37, 259-266.

[25] Seraj, O. and Tavakkoli-Moghaddam, R. (2009), A tabu serach method for a new bi-objective open shop scheduling problem by a fuzzy multi-objective decision making

approach, *International Journal of Engineering, Transaction B:* Application, 22, 269-282.

[26] Sha, D.Y. and Hsu, C.Y. (2008), A new particle swarm optimization for the open shop scheduling problem, *Computers & Operations Research*, 35, 3243-3261.

[27] Strusevich, V.A. (2000), Group technology approach to the open shop scheduling problem with batch setup times, *Operations Research Letters*, 26, 181-192.

[28] Torabi, S.A. and Hassini, E. (2008), An interactive possibilistic programming approach for multiple objective supply chain master planning, *Fuzzy Sets and Systems*, 159,193-214.

Table 6. Values of positive and negative ideal solutions and the computational times (sec.)

Sample example	$Z_1=\bar{T}_w$			$Z_2=\bar{C}_v$		
	PIS	NIS	Time	PIS	NIS	Time
6×2-a	312.71	370.67	11	298.14	352.18	5
6×2-b	154.27	204.1	26	333.15	376.4	21
6×2-c	128.25	222.13	25	397.82	551.15	18
6×2-d	87.23	136.6	49	346.05	361.95	31
5×3-a	225	229.71	107	349.64	365.18	79
5×3-b	139.32	223.54	187	314.83	405.94	101
5×3-c	102.57	151.86	301	315.65	376.56	219
5×3-d	109.11	170.44	461	310.87	388.75	415
4×4-a	229.5	252.63	432	268.07	309.71	387
4×4-b	153.59	193.41	425	268.47	386.47	407
4×4-c	207.6	226.4	392	366.14	414.21	354
4×4-d	115.67	132.73	545	138	180.45	507

Table 7. Computational results and the computational times (sec.) of example 4×4-d

γ	$\theta_1 = 0.5$, $\theta_2 = 0.5$					$\theta_1 = 0.7$, $\theta_2 = 0.3$				
	Z_1	Z_2	μ_1	μ_2	Time	Z_1	Z_2	μ_1	μ_2	Time
0	117.4	166	0.8986	0.3404	499	117.4	166	0.8986	0.3404	434
0.1	124.06	150	0.5082	0.7173	456	117.4	166	0.8986	0.3404	507
0.2	124.06	150	0.5082	0.7173	745	117.4	166	0.8986	0.3404	635
0.3	124.06	150	0.5082	0.7173	480	117.4	166	0.8986	0.3404	518
0.4	124.06	150	0.5082	0.7173	588	117.4	166	0.8986	0.3404	546
0.5	124.06	150	0.5082	0.7173	711	117.4	166	0.8986	0.3404	623
0.6	123.53	156.45	0.5275	0.5653	635	117.4	166	0.8986	0.3404	642
0.7	123.53	156.45	0.5275	0.5653	807	117.4	166	0.8986	0.3404	785
0.8	123.53	156.45	0.5275	0.5653	841	123.53	156.45	0.5275	0.5653	804
0.9	123.53	156.45	0.5275	0.5653	755	123.53	156.45	0.5275	0.5653	741
1	123.53	156.45	0.5275	0.5653	903	123.53	156.45	0.5275	0.5653	873
$\beta = 0.8$ $T = 0.4$ $R = 0.2$										

Table 8. Computational results of the quality metric

Quality Metric	Test1	Test 2	Test 3	Test 4	Test 5
PMOPSO	4	4	2	3	5
	4	1	3	7	5
	4	5	1	6	1
	2	4	1	4	3
	3	4	7	6	2
	4	3	2	4	4
	5	6	3	1	2
	5	2	4	2	2
	2	2	3	5	3
	1	4	1	2	4
CMOPSO	4	4	1	3	2
	3	2	1	2	3
	1	3	2	1	2
	0	1	2	2	3
	3	2	1	2	1
	5	5	3	1	3
	3	1	3	6	0
	3	2	4	1	5
	2	4	0	3	3
	2	0	2	2	2

Table 9. Computational results of the diversity metric

Diversity Metric	Test1	Test 2	Test 3	Test 4	Test 5
PMOPSO	326.12	395.28	461.43	387.49	801.15
	484.33	362.89	418.87	610.20	512.98
	421.24	510.92	585.86	347.67	637.03
	85.33	371.45	243.81	260.12	595.08
	471.29	402.18	677.26	651.08	680.45
	270.83	391.04	624.82	548.31	690.49
	498.16	587.88	532.21	535.09	870.59
	421.24	453.82	486.06	596.64	589.54
	326.12	475.80	537.56	573.63	418.75
	85.33	385.28	645.05	608.01	948.07
CMOPSO	305.27	525.89	612.58	411.31	530.84
	244.87	431.04	352.59	470.30	789.90
	258.62	446.19	210.97	577.14	738.95
	72.29	307.01	567.02	374.81	817.02
	252.28	361.98	564.82	111.07	379.98
	306.40	276.13	74.02	460.10	792.92
	428.16	353.75	400.90	565.26	362.39
	326.32	389.80	318.90	338.07	867.85
	254.30	450.86	512.88	602.01	488.33
	283.22	304.22	376.03	144.43	507.98

Table10. Computational results of the spacing metric

Spacing Metric	Test1	Test 2	Test 3	Test 4	Test 5
PMOPSO	0.49	0.61	2	0.45	1.15
	0.19	0.09	3	2	2.59
	2.87	2	0.85	0.9	1.6
	0.49	0.26	1.95	0.61	3
	2.45	0.99	0.62	2.21	1.12
	0	0.43	0.88	2	0.93
	1.3	0.83	2	0.68	3
	0.19	2.51	0.79	1.15	2.88
	0.54	0.69	2.45	1.2	2
	2	1.44	3	0.98	2
CMOPSO	0.81	2.85	0.66	0.57	2.96
	2	2.24	2.78	1.44	3
	2.88	2	3	1.52	2.6
	0.84	3	0.67	2	3
	0.54	1.19	3	2.88	3
	3	0.81	1.57	2	1.16
	0.85	3	2	0.92	3
	1.31	2	0.81	2.23	0.59
	1.1	1.29	0.84	0.78	2
	3	1.79	3	1.96	0.58

Table 11. ANOVA result for the quality metric

Source of Variation	DF	SS	MS	F_0	*P*-Value
Method	1	26.01	26.01	11.14	0.001
Test Problem	4	8.66	2.16	0.93	0.452
Interaction	4	3.34	0.83	0.36	0.838
Error	90	210.10	2.33		
Total	99	248.11			

Table 12. ANOVA result for the diversity metric

Source of Variation	DF	SS	MS	F_0	*P*-Value
Method	1	152245	152245	7.67	0.007
Test Problem	4	1254343	313586	15.80	0.000
Interaction	4	23627	5907	0.30	0.879
Error	90	1786034	19845		
Total	99	3216248			

Table 13. ANOVA result for the spacing metric

Source of Variation	DF	SS	MS	F_0	*P*-Value
Method	1	5.13	5.13	6.53	0.012
Test Problem	4	7.93	1.98	2.52	0.046
Interaction	4	2.88	0.72	0.92	0.457
Error	90	70.74	0.78		
Total	99	86.70			

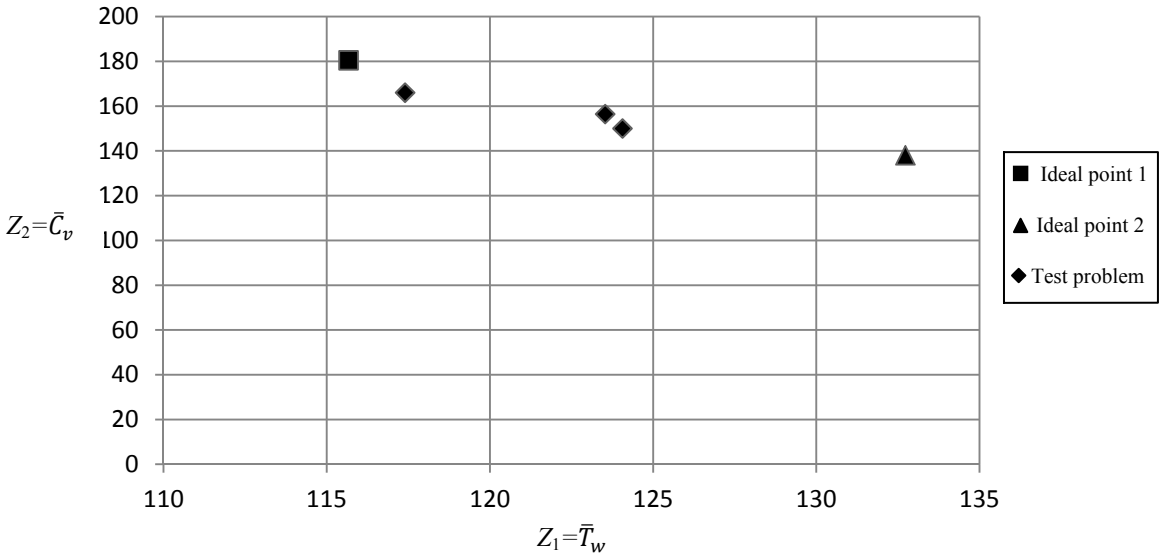

Figure 8. Pareto-optimal frontier of example 4×4-d

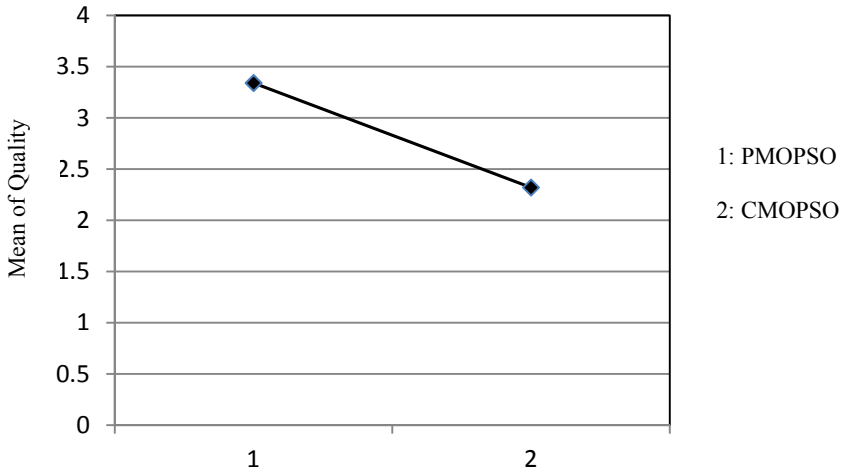

Figure 9. Effect of the solution method for the quality metric

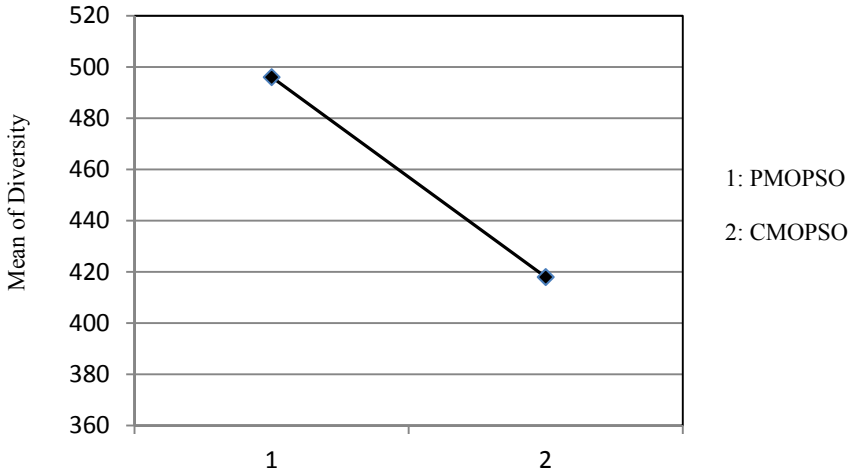

Figure 10. Effect of the solution method for the diversity metric

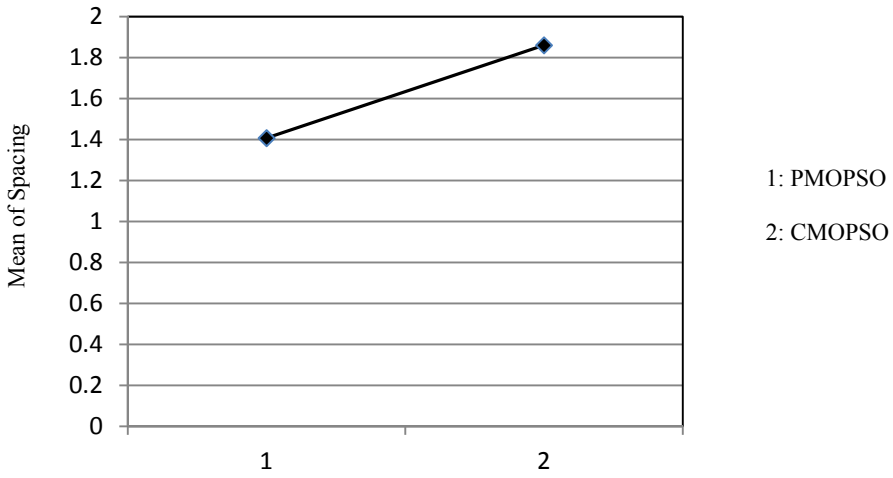

Figure 11. Effect of the solution method for the spacing metric