A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem

S.A. Torabi a, *, N. Sahebjamnia a, S.A. Mansouri b, M. Aramon Bajestani a

a School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
b Brunel Business School, Brunel University, Uxbridge, Middlesex, UK

A R T I C L E   I N F O

Article history:
Received 21 November 2012
Received in revised form 27 June 2013
Accepted 30 July 2013
Available online 30 August 2013

Keywords:
Fuzzy mathematical programming
Unrelated parallel machine scheduling
Secondary resource constraints
Multiple objective particle swarm optimization

A B S T R A C T

This paper proposes a novel multi-objective model for an unrelated parallel machine scheduling problem considering inherent uncertainty in processing times and due dates. The problem is characterized by non-zero ready times, sequence and machine-dependent setup times, and secondary resource constraints for jobs. Each job can be processed only if its required machine and secondary resource (if any) are available at the same time. Finding optimal solution for this complex problem in a reasonable time using exact optimization tools is prohibitive. This paper presents an effective multi-objective particle swarm optimization (MOPSO) algorithm to find a good approximation of Pareto frontier where total weighted flow time, total weighted tardiness, and total machine load variation are to be minimized simultaneously. The proposed MOPSO exploits new selection regimes for preserving global as well as personal best solutions. Moreover, a generalized dominance concept in a fuzzy environment is employed to find locally Pareto-optimal frontier. Performance of the proposed MOPSO is compared against a conventional multi-objective particle swarm optimization (CMOPSO) algorithm over a number of randomly generated test problems. Statistical analyses based on the effect of each algorithm on each objective space show that the proposed MOPSO outperforms the CMOPSO in terms of quality, diversity and spacing metrics.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Parallel machine scheduling problem (PMSP) is concerned with allocating a set of jobs to a number of parallel machines in order to meet customer’s requirements. In the literature, the studies on PMSP can be generally classified into the three categories [1]: identical, uniform and unrelated parallel machine scheduling problem. Among these categories, unrelated PMSP (UPMSP) represents a generalization of the other two categories in which different machines perform the same function but have different processing capabilities or capacities. However, dealing with real-life UPMSPs is a major challenge for researchers and practitioners, due not only to the fact that they are mostly NP-hard except for the objective of minimizing flow time (see [2]), but also more importantly to their special characteristics/requirements in practice. This paper focuses on an UPMSP, which has been addressed much less than the identical and uniform PMSPs in the literature especially when setup times are taken in account (see [3,4]).

Kamath [5] presents a survey on UPMSP involving makespan considerations. Li et al. [6] and Chyu and Chang [7] examine the UPMSP to minimize the mean flow time, while Cao et al. [8] study the same problem by considering the total cost functions. Lin et al. [9] address the problem to minimize the total tardiness by including the machine-dependent sequence-dependent setup times.

Researchers have addressed the more practical versions of UPMSPs by considering other features of real scheduling problems such as secondary resource constraints, non-zero ready times and so on. Chen [10] develops a heuristic method to minimize makespan in an UPMSP with different die types as a secondary resource constraint. Chen and Wu [11], and Chen [12], solve an UPMSP with auxiliary equipment constraints as secondary resource constraints. Lamothé et al. [13] propose a new model in order to minimize total tardiness by considering specific constraints such as secondary resources. As another feature, Bang [14] develops an algorithm for UPMSP with sequence-dependent set-ups and distinct ready times while minimizing total tardiness. Chen [15] develops an iterated local search to minimize the total weighted number of late jobs on UPMSP without preemption, with sequence dependent setup times and ready times.

Chang et al. [16] have proven that a single-machine, total weighted tardiness minimization problem with static job release and static machine availability and weights of all jobs being equal, is strongly NP-hard. Clearly, the single machine case considered by them is a special case of the sequence-dependent unrelated parallel machine-scheduling problem considered in this paper. Therefore, the problem investigated in this paper is also strongly NP-hard.
Amiri and Khammohammadi [17] classified the proposed methods for solving such problems into two categories as classic and intelligent algorithms (IA). While the Dynamic programming [18] and Lagrangian relaxation [19] are classified as former categories, genetic algorithm (GA) [20], PSO [21], ant colony optimization (ACO) [22], neural network (NN) [23–26] and various hybrid IAs [27] are categorized as the later one.

Thus, it is unlikely that a polynomial-time algorithm could be developed capable of determining an optimal solution for such UPMSPs like our concerned problem in practice. Hence, many researchers usually apply metaheuristic methods to deal with such problems (see e.g. [28–32]). Among them, Vallada and Ruiz [28] develop a genetic algorithm including a fast local search and a local search enhanced crossover operator. Two versions of the algorithm are proposed after extensive calibrations by using the design of experiments (DOE) approach. They conclude that their methods show an excellent performance when evaluating them over a comprehensive benchmark set of instances. Bozorgirad and Logendran [29] address a sequence-dependent group-scheduling problem on a set of unrelated-parallel machines where the run time of each job differs on different machines. They developed a meta-heuristic algorithm based on Tabu search that could find solutions at least as well as CPLEX but in drastically shorter computational time. Chen [30] considers unrelated parallel machine scheduling with sequence-dependent setup times and unequal ready times aiming to minimize the weighted number of tardy jobs. He proposes an Iterated hybrid meta-heuristic algorithm, which begins with effective initial solution generators to generate initial feasible solutions; then, hybrid meta-heuristics are applied to improve the initial solutions, integrating the principles of the variable neighborhood descent approach and tabu search. Arnaout et al. [31] introduce an enhanced ant colony optimization (ACO) Algorithm and compare its performance to other existing algorithms including ACO I, MetaRaPS, and simulated annealing (SA) on unrelated parallel machines with sequence-dependent setup times. Recently, Ruiz-Torres et al. [32] present a new unrelated parallel machine-scheduling problem with deteriorating effect and the objective of makespan minimization. They design a set of list scheduling algorithms and simulated annealing meta-heuristics and the effectiveness of these approaches are evaluated by solving a large number of benchmark instances.

Motivated by a real case study in a wire and cable manufacturer, this research deals with scheduling of a number of jobs on an unrelated parallel machine system with secondary resource constraints in which each job can only be processed if its required machine and other secondary resources (e.g. labor, tools, etc.) are available. Each job has a due date and requires a single operation with non-zero ready time. Moreover, when different jobs received from customers compete for the same resource (i.e. a set of unrelated parallel machines), in addition to agreeing on a specific due date it is customary to specify a weight or degree of importance based upon the job and the kind of relationship that exists between the customer and the producer. In addition, the setup time required for a job on a machine is dependent upon the degree of similarity or dissimilarity that exists between this job and its immediately preceding job. Thus, the problem we consider here is a sequence-dependent UPMSP aiming to minimize the total weighted flow time, the total weighted tardiness, and the machine load variation of all jobs released during the planning horizon simultaneously.

The main contributions of this paper can be highlighted as follows:

- Proposing a new fuzzy UPMSP model addressing the non-zero ready times, sequence and machine-dependent setup times, and secondary resource constraints for jobs simultaneously to cope with imprecise/ambiguous nature of critical input data as well as some real constraints in practice.
- Proposing a novel MOPSO solution method with a new solution representation modification procedure to reduce the cost of algorithm by joining the secondary resource and machine constraints’ presentations, and obtaining discrete permutation from a continuous representation.
- Exploiting new selection regimes within the proposed MOPSO for preserving gbest as well as pbest solutions.
- Employing a generalized dominance concept in a fuzzy environment to find locally Pareto-optimal frontier.
- Applying a fuzzy distance measure for calculating distance between fuzzy completion times and due dates in order to schedule jobs as close as possible to their due dates.
- Studying the effect of two competent methods (i.e. MOPSO and a conventional MOPSO called CMOPSO) and objective spaces (test problems) on the performance metrics through a novel statistical analysis.

The remainder of this paper is organized as follows. The relevant literature in UPMSP is more elaborated in Section 2 by considering the uncertainty issue. In Section 3, we define our notation, state our assumptions and propose a new fuzzy mixed-integer non-linear programming model for the proposed UPMSP problem. An introduction to particle swarm optimization along with particle swarm optimization-based scheduling literature is given in Section 4. After presenting appropriate solution representation and developing a new formulation to obtain distance between two arbitrary fuzzy numbers, we propose a novel particle swarm optimization algorithm based on the new selection regimes to solve the developed fuzzy UPMSP model in Section 5. The proposed MOPSO is validated through two classes of numerical experiments in Section 6. Performance of the proposed MOPSO is compared against a conventional MOPSO (CMOPSO) over a number of randomly generated test problems in Section 7. Finally, some concluding remarks and further research directions are provided in Section 8.

2. PMSM under uncertainty

The literature review reveals that majority of previous research in this area has focused on developing heuristic methods to find acceptable solutions in a reasonable time relying on the following assumptions:

- All of input parameters (e.g. processing times and due dates) are deterministic;
- Decision variables are deterministic such as jobs’ completion times;
- Machines are the only limited resource when processing the jobs.

There are two major drawbacks for making deterministic assumptions: (1) in many real cases, there is no enough historical data for uncertain parameters, thus, we can rarely obtain the actual value of these parameters and (2) due to inherent imprecision (ambiguity) in the input data, we cannot measure their exact values. Thereby, because of incompleteness and/or unavailability of required data, it seems to be more realistic to consider inherent imprecision/fuzziness in the critical parameters (such as processing times and due dates). Similarly, many researchers have applied fuzzy based approaches to deal with real empirical or practical application (see e.g. [33–45]). In this way, an imprecise processing time with some tolerance values (e.g. a processing time as 40 ± 5 time units) could be modeled as a fuzzy number representing the incompleteness and imprecision of required information.
Mula et al. [46] review some of the existing works in the literature of production planning under uncertainty and prove that models for production planning which do not recognize the uncertainty can be expected to generate inferior planning decisions as compared to models that explicitly account for the uncertainty. They present a general classification of different approaches that have been proposed to cope with different forms of uncertainty such as environmental uncertainty and system uncertainty. In the context of PMSP, environmental uncertainty is related to uncertainties in due dates, while system uncertainty includes the uncertainties within the processing times.

To cope with uncertainty issue in the context of scheduling, several fuzzy models involving the imprecise parameters have been developed in the literature. Among them, Peng and Liu [47] develop three fuzzy scheduling models: expectation scheduling model, α-scheduling model, and the most credible scheduling model for PMSM with fuzzy processing times. They propose a hybrid intelligent algorithm to solve the proposed models where fuzzy makespan, fuzzy lateness and fuzzy idleness are minimized. Anglani et al. [48] propose a robust approach to minimize total setup cost in PMSP with sequence-dependent setups. Then, a fuzzy mathematical programming model is formulated to provide optimal solution as a trade-off between total setup cost and the necessity degree.

To the best of our knowledge, an UPMS with: non-zero ready times, sequence and machine-dependent setup times, and secondary resource constraints in a fuzzy environment has not been studied before. This paper proposes a fuzzy multi-objective model based on these assumptions to minimize the aforementioned three performance criteria and develops a MOPSO solution method with some new features, which are elaborated in the following sections.

3. Problem formulation

We address an unrelated parallel machine scheduling problem with non-zero ready times, job sequence- and machine-dependent setup times, and the auxiliary resource constraints in a fuzzy environment. Minimizing total weighted flow time, total weighted tardiness, and machine load variation are considered as optimization objectives. Weighted flow time is a measure, which reflects work-in-process holding cost. Considering positive correlation between weighted flow time and cycle time, minimizing the former reduces the latter. Furthermore, total weighted tardiness acts as an implicit cost function. Finally, minimizing machine load variation results in a smooth schedule. Considering these three objectives simultaneously can enhance the robustness of obtained solutions.

3.1. Problem formulation

The following assumptions are considered for the problem formulation:

- Each job requires an operation that can be done on all machines,
- Jobs’ setup times are sequence and machine dependent,
- Jobs may have different arrival (ready) times,
- Assignment of a job to a machine is permitted if the required secondary resource(s) (e.g. tool, die) is (are) available,
- Processing times of jobs are machine dependent,
- Preemption and machine breakdown are not allowed,
- Processing times and due dates of jobs due to possible fluctuations in real world are subject to epistemic uncertainty (i.e. lack of knowledge in estimating these parameters precisely) and therefore are treated as possibilistic data for each of which a possibility distribution is formulated in the form of a triangular fuzzy number (by relying on available objective data and mainly subjective opinions and knowledge of experts).

The indices, parameters and variables used to formulate the problem mathematically are described below.

Indices

- \( i \) job indices \( (i = 1, \ldots, N) \)
- \( k \) machine index \( (k = 1, \ldots, M) \)
- \( g \) required secondary resource index \( (g = 1, \ldots, G) \)

Parameters

- \( a_i \) fuzzy due date of job \( i \)
- \( r_i \) ready time of job \( i \)
- \( w_i \) priority (relative importance) of job \( i \)
- \( b_{ij} \) fuzzy processing time of job \( i \) on machine \( k \)
- \( s_{ijk} \) setup time to switch from job \( i \) to job \( j \) on machine \( k \)
- \( S_g \) set of jobs that require the secondary resource \( g \)
- \( M \) an arbitrary big number

Decision variables

- \( \tilde{C}_i \) completion time of job \( i \)
- \( C_{\text{max}} \) total completion time
- \( \tilde{C}_{\text{max}} \) largest completion time on machine \( k \)
- \( T_i \) tardiness of job \( i \)
- \( F_i \) flow time of job \( i \)

\[
X_{jk} = \begin{cases} 1 & \text{if job } i \text{ precedes } j \text{ on machine } k \\ 0 & \text{otherwise} \end{cases}
\]

\[
Y_{ik} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } k \\ 0 & \text{otherwise}. \end{cases}
\]

\[
Z_{ij} = \begin{cases} 1 & \text{if processing of job } i \text{ is finished before processing of job } j \text{ starts} \\ 0 & \text{otherwise}. \end{cases}
\]

3.2. Mathematical model

According to the aforementioned assumptions and notations, the concerned problem can be formulated as the following fuzzy mixed-integer non-linear programming (FMINLP) model:

Problem P:

Minimize

\[
Z_1 = \sum_{i=1}^{N} w_i F_i = \sum_{i=1}^{N} w_i (\tilde{C}_i - r_i) \tag{1}
\]

\[
Z_2 = \sum_{i=1}^{N} w_i T_i = \sum_{i=1}^{N} w_i D(\tilde{C}_i, \tilde{a}_i) \tag{2}
\]

\[
Z_3 = \sum_{k=1}^{M} D(\tilde{C}_{\text{max}}, \tilde{C}_{\text{max}}) \tag{3}
\]

subject to:

\[
\sum_{k=1}^{M} X_{jk} = 1; \quad \forall j
\]

\[
\sum_{k=1}^{M} X_{ik} = Y_{ik}; \quad \forall j, k
\]

\[
\sum_{j=1}^{N} X_{ik} \leq Y_{ik}; \quad \forall i = 1, \ldots, N
\]

\[
\tilde{C}_j + M'(1 - X_{ijk}) \geq \text{Max}(\tilde{C}_i + s_{ijk}, r_j) + \tilde{b}_{jk}; \quad \forall i, j, k
\]

\[
\tilde{C}_j + M'(1 - Z_{ij}) \geq \text{Max}(\tilde{C}_i, r_j) + \sum_{k=1}^{M} \tilde{b}_{jk} Y_{jk}; \quad \forall i, j \in S_g, \quad i \neq j \tag{8}
\]
\[ Z_j + Z_k = 1; \quad \forall i, j \in S_k, \quad i \neq j \]
\[ \sum_{k=1}^{M} X_{ijk} \leq Z_j; \quad \forall i, j \in S_k, \quad i \neq j \]
\[ T_j \geq (\tilde{C}_i - \tilde{d}_i); \quad \forall i \]
\[ C_j \geq 0; \quad T_j \geq 0; \quad \forall i, j, k. \]

The considered objective functions, i.e., total weighted tardiness \((Z_j)\), total weighted tardiness \((Z_k)\), and machine load variation \((Z_l)\) are given by Eqs. (1)-(3), respectively. Eq. (4) ensures that each job is only assigned to one position on a machine. Eq. (5) indicates that if job \(j\) is assigned to machine \(k\), then it is followed by another job including dummy job \(0\). Constraint (6) stipulates that at most one job can immediately follow the previously assigned job \(i\) on machine \(k\). Constraint (7) calculates the completion time of job \(j\) when it is processed immediately after job \(i\) on machine \(k\). Constraints (8) and (9) guarantee that if jobs \(i\) and \(j\) require the same tool, one of them must be completed before starting the other. Constraint (10) establishes the relation between \(X_{ijk}\) and \(Z_j\). Eq. (11) calculates tardiness of job \(i\). Finally, Constraint (12) shows the non-negativity and integrality constraints.

4. Particle swarm optimization-based scheduling

With the development of evolutionary computing, Kennedy and Eberhart [49] proposed a search technique, named particle swarm optimization (PSO) for optimization problems. Like GA, PSO is also a population-based stochastic optimization algorithm, inspired by the social behavior of birds and insects. In PSO, each particle flies like a bird through the solution space of the optimization problem following the particles who have found the best solutions so far. Each particle in a swarm determines its velocity based on its own past experience. The experience is derived from both the particle as an individual and as a member of the entire population. Velocity of each individual is updated with reference to: the best individual found so far by the swarm (gbest), and the best individual found previously by that individual (pbest). Each particle is commonly represented by three \(d\)-dimensional vectors: position denoted by \(\tilde{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})\), velocity denoted by \(\tilde{v}_i = (v_{i1}, v_{i2}, \ldots, v_{id})\), and the best personal position denoted by \(\tilde{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})\). Particles have also knowledge of the best-visited global position denoted by \(\tilde{p}_g = (p_{g1}, p_{g2}, \ldots, p_{gd})\). Velocity and position vectors of the particles in the solution space are updated according to the following equations:

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

\[ x_{id}(t + 1) = x_{id}(t) + v_{id}(t + 1) \]

Shi and Eberhart [50] first introduced the inertia weight \(w\), which serves as memory of previous velocities so that large inertias favor exploration while small ones favor exploitation. Cognitive and social factors are denoted by \(c_1\) and \(c_2\) which control the influence of \(p_{best}\) and \(g_{best}\) on the search process, and \(r_1\) and \(r_2\) show random numbers generated according to the uniform distribution \(U(0,1)\).

The information sharing approach in PSO differs with other population-based algorithms. PSO does not apply the filtering operations such as crossover and/or mutation, and the information is only shared by a \(g_{best}\) or a local \(l_{best}\) [51]. Hsiao et al. [52] proposed an algorithm based on the Einstein’s general theory of relativity, which utilized the concept of gravitational field to search for the global optimal solution for a given problem. Ding et al. [53] by adding both historical information and defining an adaptive mass enhance the global search ability and accelerate convergence of the simple central force optimization (SCFO) algorithm. Li et al. [54] utilized two critical operations including the self-update of reference-group and the interactive-update process between the reference-group and floating-group to find global optimum solutions. The selection of social and cognitive leaders \((g_{best} \text{ and } p_{best})\) is the key point in multi-objective particle swarm optimization. The first attempt in applying particle swarm to multi-objective optimization was done by Moore and Chapman [55]. They modified the \(p\)-vector of the particles so that each particle keeps track of all non-dominated solutions (using Pareto preference) experienced by itself.

Goh et al. [56] adapt a competitive and cooperative co-evolutionary approach for multi-objective PSO algorithm design, which appears to have considerable potential for solving complex optimization problems by explicitly modeling the co-evolution of competing and cooperating species. Qasem and Shamsuddin [57] introduce a multi-objective PSO that evolves toward Pareto-optimal front defined by several objective functions with model accuracy and complexity. Their proposed approach is simple with faster convergence to Pareto-optimal solutions. It has two main advantages: first, the proposed method can be applied to any real-world problem and second, it shows better performance in terms of error, sensitivity, specificity and accuracy for benchmark data sets. Jia et al. [58] improve multi-objective PSO by employing a novel diversity preservation strategy that combines the information of distance and angle into similarity judgment to select global best and thus the convergence and diversity of the Pareto front is guaranteed. In this manner, they can distribute enough Pareto solutions in the Pareto front evenly. Wang et al. [59] propose a novel hybrid multi-objective PSO based on Baldwinian learning mechanism that improves the local search ability of PSO, and use the Pareto dominance and crowding distance to update the solutions.

Due to the simple concept, easy implementation, and quick convergence, PSO has been widely applied in a variety of optimization problems. However, because of the specific algorithmic structure of PSO (updating of position and velocity of particles in a continuous manner), few studies of PSO have been reported in the area of scheduling problems. In order to apply PSO to the scheduling problems successfully, several researchers have extended PSO or designed the encoding schemes for converting the continuous position values into the discrete job sequences. Zhang et al. [60] develop PSO algorithm to solve the multi-objective job-shop scheduling problem with several conflicting and incomparable objectives. They show PSO which integrates local search and global search scheme possesses high search efficiency. Moslehi and Mahnam [61] present a new approach based on a hybridization of the PSO and local search algorithm to solve the multi-objective job-shop scheduling problem. Niu et al. [62] design a new solution representation based on real number encoding, which can conveniently convert the job sequences to continuous position values. In order to enhance the performance of PSO, they introduce the Clonal Selection Algorithm (CSA) into the PSO thereby proposing a new CSPSO method. Shao et al. [63] propose a hybrid discrete PSO to identify an approximation of the Pareto front for flexible job-shop scheduling problem.

5. The proposed MOPSO

In this section, we propose a posteriori optimization approach, i.e. a MOPSO algorithm to solve the problem. Noteworthy, in a posteriori multi-objective approach, which is also known as generation approach, a number of efficient compromise solutions are first generated to estimate the Pareto-optimal frontier without using of any preference information provided by the decision maker. In this way,
a set of Pareto-optimal solutions is provided for the decision maker by which she or he will then be able to select her/his final preferred compromise solution based upon her/his preferences.

5.1. Solution representation

One of the most important decisions when designing a meta-heuristic lies in deciding how to represent solutions and relate them in an efficient way to the searching space. Representation should be easy to decode in order to reduce the cost of the running of algorithm. Two different kinds of solution representations are used simultaneously in this research, i.e. the well-known job-to-position and continuous representation. Therefore, each particle would have a job-to-machine and a continuous representation concurrently, each of which is used in different steps of our algorithm. In the next sections, we discuss how they are used:

5.1.1. Job-to-position representation

One of the most widely used representations for scheduling problems is job-to-position representation [64]. In this kind of representation, a single row array with the size equal to the number of jobs is first formed. Since, for processing each job, the machine may need a secondary resource simultaneously, we modify the job-to-position representation in which the first row represents the location of jobs in the sequence, the next k rows show the assigned jobs to k parallel machines and the last g rows represent the priority of each job on each secondary resource. A typical job-to-position representation for k machines and g secondary resources has been shown in the Fig. 1. However, decoding this type of solution representation will raise the cost of the PSO algorithm considerably. Hence, in the next section, we take into account the continuous nature of PSO algorithm to cope with this issue.

5.1.2. Modified continuous representation

Many researchers have used the representation proposed by Tasgetiren et al. [64] for scheduling problems with continuous values. In an UPMS, a continuous representation is an array consisting of n real values. These values indicate a particle’s position in each dimension of the n-dimensional space they move in. However, the upper bound for these values limits the maximum distance that each particle is allowed to move in each iteration. In this paper, the upper bound is set to 4 while each particle could move between [0,4]. Notably, the use of hard bounds on distance that each particle is allowed to move in each iteration, presents some problems. Obviously, the best value of this bound is problem-specific, but no reasonable rule of thumb is known so far [49]. However, in our problem, maximum distance greater than 4 would lead to growth in the oscillations of the particles, which might eventually leave the region of interest in the search space. Since the position of a particle does not represent a solution throughout the algorithm, a transformation is needed to obtain a permutation from a continuous representation and vice versa. By using a solution representation modification procedure we try to: (1) simply represent a solution and (2) convert a discrete permutation to an equivalent continuous representation and vice versa. The proposed solution representation modification procedure is as follows:

1. Form the initial representation (i.e. $X_{k\times n}$ and $T_{g\times n}$ matrices) as shown in Fig. 1.
2. Sort the job position values in the $X_{k\times n}$ matrix in ascending order for each machine,
3. Sort the job position values in the $T_{g\times n}$ matrix in ascending order for each secondary resource,
4. Sort the jobs by considering the steps 2 and 3 where comparing of genes shows the priorities of jobs in using common unique tools. If two jobs with common tool are assigned to the same machine, the preceding job is given priority.

For instance, consider an example with five jobs, two machines and two secondary resources. Those jobs requiring the secondary resource g are as: $S_{g=1} = \{3, 5\}$ and $S_{g=2} = \{1, 2, 4\}$. To show the continuous representation, assume that 0.2, 2.2 and 3.7 have been assigned to job numbers 1, 3 and 4 on the first machine and 1.6 and 1.9 to job numbers 2 and 5 on the second machine, respectively as shown in Fig. 2. Based on the proposed procedure, the job position values at each row are first sorted in an ascending order and then the final sequence of jobs on each machine is obtained. Accordingly, the discrete counterpart of above continuous representation has been shown in Fig. 3.

5.2. Fuzzy distance

Here, the pattern of triangular possibility distribution is adopted to represent each fuzzy parameter (i.e. fuzzy processing times and due dates) in the model. Noteworthy, the possibility distribution of fuzzy processing times and due dates have been derived based on some available objective (historical) data as well as subjective data pertaining to the knowledge and experiences of decision maker(s).

In the proposed fuzzy UPMSP, since the processing times and due dates are considered as fuzzy numbers, we reformulate the objective functions (1–3) based on the concept of fuzzy distance. Let $\tilde{A} = (a^1, a^2, a^3)$ and $\tilde{B} = (b^1, b^2, b^3)$ be two triangular fuzzy parameters. It is well known that the summation operator works according to the following formula:

$\tilde{A} + \tilde{B} = (a^1 + b^1, a^2 + b^2, a^3 + b^3)$ (15)

Fig. 1. A job-to-position representation.

Fig. 2. Sample set of random number.

Fig. 3. Discrete representation of solution.
According to the Zadeh’s extension principle, the membership function for the maximum of two fuzzy numbers can be obtained by the following formula:

$$\mu_{\lambda \wedge \beta}(z) = \sup_{z \in \lambda \wedge \beta} \{\min[\mu_{\lambda}(x), \mu_{\beta}(y)]\}$$  

where $\lor$ shows the max operation. Based on the extension principle, it can be proved that the result of max operation is always a fuzzy number but not necessarily with a triangular pattern. For simplicity in data acquisition, we use an approximation of max formula introduced by Sakawa and Kubotu [65] as follows:

$$\lambda \lor \beta = (a_1, a_2, a_3) \lor (b_1, b_2, b_3) \equiv (a_1 \lor b_1, a_2 \lor b_2, a_3 \lor b_3)$$  

The max and sum operations are utilized to calculate the fuzzy starting and completion times of each job, respectively. Let us provide the following parametric definitions.

**Definition 1.** A fuzzy number $u$ in parametric form is a pair $(\bar{u}(r), \tilde{u}(r))$, $0 \leq r \leq 1$ that satisfy the following requirements:

1. $\bar{u}(r)$ is a bounded increasing left continuous function,
2. $\tilde{u}(r)$ is a bounded decreasing left continuous function,
3. $\bar{u}(r) \leq \tilde{u}(r)$, $0 \leq r \leq 1$

The membership function for $u=(x_0, \sigma, \beta)$, a triangular fuzzy number with one defuzzifier $x_0$, left fuzziness $\sigma > 0$, and right fuzziness $\beta > 0$, can generally be defined as:

$$u(x) = \begin{cases} 
\frac{1}{\sigma}(x-x_0)\
1\
\frac{1}{\beta}(x_0+x-\beta)
\end{cases}$$  

The parametric form of above membership function can be presented as follows:

$$y(r) = \sigma(r-1)+x_0, \bar{u}(r) = \beta(1-r)+x_0$$  

**Definition 2.** For arbitrary fuzzy numbers $u=(\bar{u}, \tilde{u})$, and $v=(\bar{v}, \tilde{v})$, the distance between $u$ and $v$ in $E$ can be calculated by using the following formula [66]:

$$D(u, v) = \left[\int_{0}^{1} (\bar{u}(r) - \bar{v}(r))^2 dr + \int_{0}^{1} (\tilde{u}(r) - \tilde{v}(r))^2 dr\right]^{1/2}$$  

**Theorem 1.** If $u=(x_{0u}, \sigma_u, \beta_u)$ and $v=(x_{0v}, \sigma_v, \beta_v)$ are two triangular fuzzy numbers, then:

$$D(u, v) = \left[\left(\frac{\sigma_u - \sigma_v}{3}\right)^2 + \left(\frac{\beta_u - \beta_v}{3}\right)^2 + (x_{0u} - x_{0v})^2 + (\beta_u - \beta_v)^2 + (\sigma_u - \sigma_v)^2\right]^{1/2}$$  

**Proof.** The proof can easily be concluded by applying Eqs. (19) and (20).

**5.3. Generalization of dominance in fuzzy environment**

In single objective optimization problems, it is easy to compare one solution to another but in multi-objective problems, a solution that is inferior to another one in one objective, may be better with respect to another objective. Under these circumstances, the concept of non-dominated solution is used.

A general multi-objective minimization problem with $p$ decision variables and $q$ objectives ($q > 1$) can be presented as:

$$\text{Min} \ y=f(x)=f_1(x), f_2(x), \ldots, f_q(x), \text{where } x \in \mathbb{R}^p, \text{and } y \in \mathbb{R}^q.$$  

A solution $x^*$ is non-dominated (Pareto-optimal) if there is no other solutions $x \in \mathbb{R}^p$ such that $f_i(x) \leq f_i(x^*)$, $\forall i$ with at least one strict inequality.

However, for our concerned problem, we are interested not to convert the original fuzzy objective vector into its equivalent crisp vector. To achieve this, we need work with fuzzy non-dominated solutions (see Koduru et al. [67] for more details about the concept of fuzzy dominance).

Notably, the most of current approaches concentrate on adapting an evolutionary algorithm to generate the Pareto frontier. Fernandez et al. [68] presented a new idea to incorporate preferences into a multi-objective evolutionary algorithm. They introduced a binary fuzzy preference relation that expresses the degree of truth of the predicate “$x$ is at least as good as $y$.” On this basis, a strict preference relation with a reasonably high degree of credibility can be established on any population. Panigrahi et al. [69] used fuzzy dominance based sorting procedure to select the Pareto optimal front for multi-objective bacterial foraging optimization technique. Lei [70] proposed multi-objective PSO to solve fuzzy job shop scheduling problem. He formed Pareto archive PSO, in which the global best position selection is combined with the crowding measure-based archive maintenance. Recently, Di Martino and Sessa [71] extended a new Fuzzy PSO algorithm to determine hotspot areas where the data are events geo-referenced as points on the geographic map.

In this work, our main objective is to estimate the Pareto-optimal frontier in a fuzzy domain. Consequently, we have developed a fuzzy Multi-objective PSO using fuzzy dominance based ranking methods. Many techniques for ranking fuzzy numbers have been proposed since 1976. Chen and Hwang [72] classified ranking methods into four major categories: preference relation, fuzzy mean and spread, fuzzy scoring, and linguistic expression. However, the centroid-based distance method suggested by Cheng [73] is adopted here due to its strong theoretical foundation and good results reported in the literature, in which fuzzy numbers are ranked based on their Euclidean distance from their centroid points to the origin. Wang et al. [74] refuted the centroid formula provided by Cheng [73] and presented the accurate form.

**Definition 3.** For a plane area $A$, bounded by two continuous curves $y = y_1(x), y = y_2(x)$ where $y_1(x) \leq y_2(x)$ and two lines $x = a, x = b$, the static moments and coordinates of the respective centroid are calculated according to the following equations:

$$M_x = \frac{1}{2} \int_a^b (y_2^2 - y_1^2) dx; \quad M_y = \int_a^b x(y_2 - y_1) dx$$  

$$x_c = \frac{M_y}{M_x}; \quad y_c = \frac{M_x}{M_x}$$  

where $x_c$ and $y_c$ are coordinates of the centroid of $A$, respectively, and $S$ denotes the area of the plane $A$.

**Theorem 2.** If $A$ is characterized by the inverse function of a triangular fuzzy number in Fig. 4, then:

$$x_c = x_0 + \frac{\beta - \sigma}{3}; \quad y_c = \frac{1}{3}$$  

**Proof.** The proof can easily be concluded by applying Eqs. (22) and (23).

Finally, the greatest associated Euclidean distance $R = \sqrt{x_c^2 + y_c^2}$ is used to rank the triangular fuzzy numbers.
5.4. Generating initial velocities

Real values are considered for velocities because of the fact that positions of particles (denoted by \([X_{k,n}]\) have continuous form. The size of the velocity matrix is the same as that of the position matrix. Initial velocity matrices are generated as follows:

\[
v_i(k) = \left( v_{\text{min}} + (v_{\text{max}} - v_{\text{min}}) \times r_1 \right)
\]

(25)

where \(v_{\text{min}} = -4, v_{\text{max}} = 4, \) and \(r_1\) is a uniform random variable between \([0,1]\). The range of acceptable velocity values (i.e. \([v_{\text{min}}, v_{\text{max}}]\)) is chosen so that \(v_i(k)\) is \(X_{\text{max}}\) where \(v_i(k)\) denotes velocity of job \(j\) on machine \(i\) for the \(k\)-th particle.

5.5. Updating positions and velocities

The procedure for updating particle velocities and positions is principally similar to single objective optimization. The difference lies in the way particle movements are updated. In this paper, each particle is allowed to have an archive consisting of its best previous positions. As new \(p_{\text{best}}\) solution are added to the archive, solutions that consequently become dominated are discarded from the archive. As a result, this set will ultimately represent an approximation of Pareto front found by that particle. Despite more complexity of this approach compared to preserving only one \(p_{\text{best}}\), enhanced search capability would compensate for the computational cost.

According to the literature, \(g_{\text{best}}\) methods can fall into two categories: unrestricted and restricted. ‘Unrestricted’ approach allows for \(g_{\text{best}}\) to be selected for a given member of swarm from anywhere in the archive. In contrast, ‘restricted’ approach constrains the possible \(g_{\text{best}}\) for a given individual by using some form of distance measure. Fieldsend [75] show that although the later group is more desirable, there are some adverse consequences associated with this approach.

Our proposed approach employs a combination of these two \(g_{\text{best}}\) categories. In order to maintain diversity of search direction and to avoid premature convergence to local optima, each \(g_{\text{best}}\) is given a chance. The employed procedure can be clarified by the following example: assume that 100 individuals and 8 \(g_{\text{best}}\) archive members exist in a given problem. The quotient of 100 :: 8 = 12 provide chance of each \(g_{\text{best}}\) in updating process of particle velocities. It is clear that some particles (100 – 12 × 8 = 4 particles) are not yet assigned a \(g_{\text{best}}\). Therefore, the \(g_{\text{best}}\) with the highest crowding distance values are given extra chance, i.e. four \(g_{\text{best}}\) are selected 13 times as a social leader, while the remaining \(g_{\text{bests}}\) will have a 12-time chance. To allocate a \(g_{\text{best}}\) to each individual, the minimal Euclidean distance approach is applied.

After determining the appropriate \(g_{\text{best}}\) for each particle, the \(p_{\text{best}}\) with maximum distance from the selected \(g_{\text{best}}\) for each particle is chosen. The reason is that each particle is pushed/pulled toward/from its \(p_{\text{best}}\) and \(g_{\text{best}}\) operating points along its current velocity; thereby generating a hypercube in the solution space. Selecting the \(p_{\text{best}}\) with the maximum distance in this situation will expand this bounded region.

Finally, the particle velocities and positions are updated using Eqs. (13) and (14). The inertia weight is updated as \(w(t) = w(t - 1) \times \beta\) where \(\beta\) is a decremental factor to control the effect of inertia weight on exploration and exploitation. In each generation, the \(g_{\text{best}}\) archive set is updated as follows. With the new solution being added, solutions that consequently become dominated are excluded from both \(p_{\text{best}}\) and \(g_{\text{best}}\) archives. Updating \(g_{\text{best}}\) archive set in each iteration in presence of three objective functions with discrete matrices of decision variables, require considerable time.

In order to reduce this time and based upon the observations made during our experimentations, we decided not to put a bound on the archive size. In other words, all non-dominated solutions are maintained in the archive.

The above procedure is implemented for a number of predetermined iterations and the final non-dominated \(g_{\text{best}}\) archive set is reported as the result. The following pseudo code summarizes main steps of the proposed algorithm:

- Initialize a population of particles with random positions,
- Generate random velocities using Eq. (25),
- Construct the personal Pareto archive for each particle to store the best individual found so far,
- Construct the global Pareto archive which is empty at the beginning,
- Repeat the following procedure for a predetermined number of iterations:
  - Perform fast-non-dominated sorting,
  - Update the global Pareto archive,
  - Determine the best global position matrix for each particle using the procedure explained in Section 3.5,
  - Determine the best personal position matrix for each particle using the procedure explained in Section 3.5,
  - Update the velocity and position of each particle using Eqs. (13) and (14), respectively.
- Update the personal Pareto archive of each particle,

Report solutions of global Pareto archive as final non-dominated solutions.

6. Experimental results

In this section, the performance of the proposed MOPSO is compared with a conventional multi-objective particle swarm optimization (CMOPSO). The difference between the proposed MOPSO and CMOPSO lies only in the selection of social and cognitive leaders. A non-dominated solution in global archive set with the highest crowding distance is considered as a \(g_{\text{best}}\) for all particles and the \(p_{\text{best}}\) for each particle is randomly chosen from the personal Pareto archive.

Before going through the details of numerical experiments, we first proof a proposition to calculate \(C_{\text{max}}\). After generating two sets of the small and large size problems, we discuss how the parameters’ tuning is performed.

6.1. \(C_{\text{max}}\) Calculation

To calculate \(C_{\text{max}}\), the following proposition is applied.

**Proposition 1.** There exists an optimal schedule, in which the sum of processing times of the jobs assigned on machine \(k\) does not exceed
\[ A_k = \frac{1}{m} \left( \sum_{j \in M} \max_{i \in j} (\max_{q \in g_i} AP_{ij}^q) + \sum_{r \in M} \max_{j \in j} (\max_{q \in g_i} AP_{ij}^q) \right) \]  

(26)

where \( AP_{ij}^q = P_{ij}^q + S_{ij}(i,j = 1, 2, \ldots, n) \)

**Proof.** Let \( l_k \) be the last job to be processed on machine \( k \), and \( C_{kl} \) be the completion time of \( l_k \) in an optimal schedule. For any other machine \( r (r \neq k) \), it must be

\[ C_{kl} + AP_{ij}^q \geq C_{kj} \quad \text{and} \quad C_{kl} \geq C_{kj} - AP_{ij}^q, \forall r \]  

(27)

With \( l_k \) having been taken from machine \( k \) and placed on machine \( r \), the total cost (i.e. the total weighted tardiness) would decrease or not change if Eq. (27) did not hold. By summing Eq. (27) over all machines \( r \neq k \), we obtain:

\[ \sum_{r \in M} C_{kl} - C_{kj} \geq (m-1)C_{kj} - \sum_{r \neq k} AP_{ij}^q \]

This implies that:

\[ C_{kl} \leq \frac{1}{m} \left( \sum_{r \in M} C_{kl} - \sum_{r \neq k} AP_{ij}^q \right) \]

Taking into account the following inequalities:

\[ \sum_{r \in M} C_{kl} \leq \sum_{r \in M} \max_{i \in j} (\max_{q \in g_i} AP_{ij}^q) \]

\[ AP_{ij}^q \leq \max_{i \in j} (\max_{q \in g_i} AP_{ij}^q) \]

We would obtain:

\[ C_{kl} \leq \frac{1}{m} \left( \sum_{r \in M} \max_{i \in j} (\max_{q \in g_i} AP_{ij}^q) + \sum_{r \in M} \max_{j \in j} (\max_{q \in g_i} AP_{ij}^q) \right) \]

For any machine \( k \), the value of \( A_k \) can represent an upper bound on the total processing times of the jobs assigned to machine \( k \). Thus, \( C_{\max} = \max_{k=1}^M A_k \).

### 6.2. Test problems

In this section and inspired by a cable and wire manufacturing company owning multiple unrelated parallel machines with sequence dependent setup times at each production stage, two separate classes of randomly generated test problems, i.e. the small and large sized instances are conducted and their numerical results are also reported separately to show the effect of the problem's size on the three important performance metrics in multi-objective problems. Noteworthy, due to confidentiality as well as the lack of some required data at our considered case study, all required data in the test problems have been generated randomly. However, the generation of these random data were carried out in such a way that they would be in the range of real data and close to them. In this way, we were care about this issue that these problem instances could capture the main characteristics of real problems and therefore could serve as proxy for real cases. It should also be noted that the kind of data generation enabled us to do more numerical tests to find more reliable statistical results as well.

Furthermore, all the required procedures are implemented using the Matlab software and executed on a Pentium 3 with 2 GHz, and Windows XP using 1 GB of RAM. For both of these test problems, we consider the following assumptions:

1. Each experiment is conducted using 12 randomly generated test problems as summarized in Table 1.
2. Processing times and due dates have symmetric triangular possibility distributions generally denoted by \( \hat{C} = (C^l, C^m, C^u) \) where \( C^l, C^u \), and \( C^m \) are the most pessimistic, the most possible, and the most optimistic values of \( C \), respectively.
3. The most possible values for processing times are generated using a continuous uniform distribution ranging from 5 to 40.
4. Ready times are chosen as \( r = U(0, \alpha) \cdot C_{\max} \), where \( \alpha \) allows us to control the job arrival.
5. The most possible values for due dates are generated as \( d_j = r_j + U(0, \beta) \cdot \min_{r=1}^{M} \left( \min_{i=1}^{J} A_{ij} \right) \) where \( \beta \) allows to control the tightness.
6. The fuzzy spread of processing times' and due dates' distributions are generated using a continuous uniform distribution ranging in the interval \([1.5]\).

### 6.3. Performance metrics

To validate the reliability of the proposed MOPSO, the following three comparison metrics are taken into account [76]:

#### 6.3.1. Number of Pareto-optimal solutions (N.P.S.)

This metric shows the number of locally Pareto-optimal solutions that each algorithm can find. It is obvious that a higher value of this metric signifies better exploration and a more diverse search direction.

#### 6.3.2. Diversity metric (D)

This metric measures the spread of the obtained solutions (i.e. how diverse the obtained solutions are) which is defined as follows:

\[ D = \sqrt{\sum_{i=1}^{n} \max(||x_i - y_i||, \hat{x}, \hat{y} \in F)} \]  

(28)

where \( F \) is the practical (estimated) tradeoff surface, and \( n \) is its respective dimension.

#### 6.3.3. Spacing metric (S)

The spacing metric allows us to measure the uniformity of the obtained points in the estimated tradeoff surface. This metric is defined as follows:

\[ S = \frac{\sum_{i=1}^{N-1} |d_i - \bar{d}|}{(N - 1) \bar{d}} \]  

(29)

where \( d_i \) is the Euclidean distance between two successive solutions in the obtained non-dominated set \( F \), and \( \bar{d} \) is the average of \( d_i \)'s. This metric measures the uniformity of the solutions' spread using a triangulation technique or a voronoi diagram approach. The above procedure can be extended to estimate spread of solutions in higher dimensions. To apply the triangulation technique in a three dimensional space, we divide the space by the Delaunay triangulation into some tetrahedrons whose vertices are the obtained non-dominated solutions. Note that the Delaunay triangulation is a set of lines connecting each point to its natural neighbors. Finally, we substitute the term \( d_i \) in the above formula by the volume of these tetrahedrons.

It should be noted that the aforementioned performance metrics considered here are widely used by other researchers and practitioners too (see for example [59,69,70]).
6.4. Parameters setting

Obviously, the performance of a PSO algorithm greatly depends on its parameters. The main parameters of a PSO that let it to be a competing algorithm at the best level, are cognitive factor \( c_1 \), social factor \( c_2 \), swarm size \( N \), number of iterations \( K \), and inertia factor \( w \). Notably, inertia factor controls the quality of global exploration and local exploitation that are both crucial in finding a good (hopefully optimal) solution efficiently.

For tuning of the proposed algorithm’s parameters appropriately, we have conducted an orthogonal experiment with different set of parameters that is a major tool in the Taguchi design. The purpose of conducting an orthogonal experiment is to determine the optimum level for each controllable parameter and to establish the relative significance of individual parameters in terms of their main effect on the results (for more detail see [77,78]).

Due to the nonlinear effect of the PSO parameters, it has been decided to use four levels for the parameters (see Table 2). Taguchi method begins with the selection of orthogonal array with distinct number of levels \((l)\) defined for each of the parameters \((f)\). The minimum number of trials \( N_{\text{min}} \) in the orthogonal array is \( N_{\text{min}} = (l-1)f+1 \). For example, for 5 parameters and 4 levels, just 16 experiments are required to study the entire PSO parameters’ tuning for each test problem as reported in Table 3.

Taguchi design uses a logarithmic function of desired response (serves as the objective function for optimization) instead of mean value to interpret the trial results in the optimum setting analysis, because it can reflect both mean and variation of the performance characteristics [70]. In this paper, Taguchi parameter design with the utility concept, which has been introduced by Kumar et al. [79] is applied. The logarithmic function associated with the responses, are given as Eqs. (30)–(32). In addition, the multi-response ratio (33) and the effect of a parameter level \( i \) on parameter \( j \) (34) are used here (see [78,79]).

\[
\eta_1 = -10 \log_{10} \left[ \frac{1}{D^2} \right] \tag{30}
\]

\[
\eta_2 = -10 \log_{10} \left[ \frac{1}{D^2} \right] \tag{31}
\]

\[
\eta_i = -10 \log_{10} \left[ \frac{1}{D^2} \right] \tag{32}
\]

\[
\eta = \sum_{i=1}^{3} W_i \eta_i \tag{33}
\]

\[
m_{i,j} = \frac{1}{T} \sum_{i=1}^{T} \eta_{i,j} \tag{34}
\]

where \( v \) is associated with the responses, and \( w_i \) is the weighting factor of metric \( v \). Finally, the optimum level of a parameter is the level, which gives the highest response. For tuning two categories of problems (i.e. small and large sized test problems), different test problems with four levels of parameters (16 trial problem) were solved and summarized in Table 4. It should be noted that due to space limitation, the logarithmic function and utility associated with the responses, has just been summarized for three trials (TN1 to TN3) and for other test problems, but they can be provided upon request. In addition the effect of a parameter level \( i \) for parameter \( j \), \( m_{i,j} \) were calculated and the results of four samples, i.e. SS–1, SS–2, LS–1 and LS–2 are reported and highlighted in Table 5. In this manner, the optimum level of parameters for all test problems are achieved. For example, for SS–1 and LS–1, the optimum level of parameters are \((4, 2, 2, 3, 4)\) and \((4, 3, 1, 4, 3)\), respectively.

### Table 2

PSO parameters and their levels.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Cognitive factor ( c_1 )</td>
<td>0.5</td>
</tr>
<tr>
<td>Social factor ( c_2 )</td>
<td>0.5</td>
</tr>
<tr>
<td>Swarm size ( N )</td>
<td>15</td>
</tr>
<tr>
<td>Number of iterations ( K )</td>
<td>200</td>
</tr>
<tr>
<td>Inertia factor ( w )</td>
<td>0.2</td>
</tr>
</tbody>
</table>

### Table 3

Trial experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Trial number (TN)</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( N )</th>
<th>( K )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Table 4
Utility associated with the response of performance metrics.

<table>
<thead>
<tr>
<th>TN</th>
<th>SS-1</th>
<th>SS-2</th>
<th>SS-3</th>
<th>SS-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3.32</td>
<td>4.07</td>
<td>3.2</td>
<td>3.32</td>
</tr>
<tr>
<td>3</td>
<td>3.77</td>
<td>3.86</td>
<td>3.05</td>
<td>3.86</td>
</tr>
</tbody>
</table>

(1) The proposed MOPSO can achieve a greater number of Pareto-optimal solutions,
(2) It provides various non-dominated solutions with more average value for the diversity metric,
(3) These data reveal that the non-dominated solutions obtained by the proposed MOPSO are more uniformly distributed when compared to the CMOPSO.

6.5. Computational results

In this section, the proposed MOPSO is applied to solve all the test problems and its performance is compared with those of obtained when using the CMOPSO. The computational results regarding the proposed MOPSO and the CMOPSO in small and large sized problems for different metrics and objective functions have been reported in the Tables 6 and 7, respectively. As illustrated in these Tables, the proposed MOPSO shows better performance than the CMOPSO in both of the problem sets.

Although the computational time of the proposed MOPSO is greater than the CMOPSO, but its superiority can be realized as follow:

Table 5
The values of $m_{ij}$

<table>
<thead>
<tr>
<th>TP</th>
<th>Level</th>
<th>c1</th>
<th>c2</th>
<th>N</th>
<th>w</th>
<th>K</th>
<th>TP</th>
<th>Level</th>
<th>c1</th>
<th>c2</th>
<th>N</th>
<th>w</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS-1</td>
<td>1</td>
<td>6.90</td>
<td>5.51</td>
<td>6.56</td>
<td>5.36</td>
<td>5.65</td>
<td>LS-1</td>
<td>1</td>
<td>16.97</td>
<td>16.72</td>
<td>16.74</td>
<td>16.69</td>
<td>16.33</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.22</td>
<td>6.60</td>
<td>7.32</td>
<td>6.83</td>
<td>6.35</td>
<td></td>
<td>3</td>
<td>15.73</td>
<td>17.61</td>
<td>15.49</td>
<td>16.51</td>
<td>16.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5.53</td>
<td>6.05</td>
<td>6.72</td>
<td>5.06</td>
<td>6.08</td>
<td></td>
<td>4</td>
<td>17.23</td>
<td>14.80</td>
<td>15.89</td>
<td>17.58</td>
<td>14.17</td>
</tr>
<tr>
<td>SS-2</td>
<td>1</td>
<td>10.51</td>
<td>9.42</td>
<td>10.83</td>
<td>11.70</td>
<td>11.70</td>
<td>LS-2</td>
<td>1</td>
<td>14.07</td>
<td>16.54</td>
<td>14.86</td>
<td>15.77</td>
<td>14.73</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>11.36</td>
<td>10.40</td>
<td>10.44</td>
<td>9.03</td>
<td>9.03</td>
<td></td>
<td>3</td>
<td>15.44</td>
<td>17.77</td>
<td>16.65</td>
<td>16.05</td>
<td>17.05</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>11.93</td>
<td>9.91</td>
<td>9.94</td>
<td>9.26</td>
<td>9.86</td>
<td></td>
<td>4</td>
<td>18.23</td>
<td>14.10</td>
<td>17.01</td>
<td>16.57</td>
<td>16.21</td>
</tr>
</tbody>
</table>

The bold values show the selected level for each parameter.
Table 6
Computational results for the small-sized problems.

<table>
<thead>
<tr>
<th>TP</th>
<th>N.P.S.</th>
<th>D</th>
<th>S</th>
<th>ACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>CM</td>
<td>PM</td>
<td>CM</td>
</tr>
<tr>
<td>SS-1</td>
<td>3</td>
<td>3</td>
<td>355.68</td>
<td>223</td>
</tr>
<tr>
<td>SS-2</td>
<td>4</td>
<td>4</td>
<td>343.04</td>
<td>265.48</td>
</tr>
<tr>
<td>SS-3</td>
<td>3</td>
<td>3</td>
<td>366.89</td>
<td>234.70</td>
</tr>
<tr>
<td>SS-4</td>
<td>5</td>
<td>5</td>
<td>315.03</td>
<td>305.72</td>
</tr>
<tr>
<td>SS-5</td>
<td>5.34</td>
<td>5.3</td>
<td>232.79</td>
<td>220.25</td>
</tr>
<tr>
<td>SS-6</td>
<td>5.76</td>
<td>5.76</td>
<td>374.02</td>
<td>346.07</td>
</tr>
<tr>
<td>SS-7</td>
<td>4.87</td>
<td>4.8</td>
<td>344.74</td>
<td>278.72</td>
</tr>
<tr>
<td>SS-8</td>
<td>5.6</td>
<td>5.6</td>
<td>235.68</td>
<td>211.1</td>
</tr>
<tr>
<td>SS-9</td>
<td>7.4</td>
<td>7.43</td>
<td>293.09</td>
<td>274.54</td>
</tr>
<tr>
<td>SS-10</td>
<td>3.84</td>
<td>3.68</td>
<td>370.33</td>
<td>296.87</td>
</tr>
<tr>
<td>SS-11</td>
<td>5.79</td>
<td>5.79</td>
<td>389.93</td>
<td>327.52</td>
</tr>
<tr>
<td>SS-12</td>
<td>6.37</td>
<td>6</td>
<td>269</td>
<td>170.11</td>
</tr>
</tbody>
</table>

PM, proposed MOPSO; CM, CMOPSO; ACT, average computational time (in s).

Table 7
Computational results for the large-sized problems.

<table>
<thead>
<tr>
<th>TP</th>
<th>N.P.S.</th>
<th>D</th>
<th>S</th>
<th>ACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>CM</td>
<td>PM</td>
<td>CM</td>
</tr>
<tr>
<td>LS-1</td>
<td>7.81</td>
<td>6.81</td>
<td>655.68</td>
<td>651</td>
</tr>
<tr>
<td>LS-2</td>
<td>9.23</td>
<td>8.23</td>
<td>643.04</td>
<td>641.94</td>
</tr>
<tr>
<td>LS-3</td>
<td>9.94</td>
<td>8.94</td>
<td>666.89</td>
<td>542.93</td>
</tr>
<tr>
<td>LS-4</td>
<td>10.65</td>
<td>9.65</td>
<td>615.03</td>
<td>492.39</td>
</tr>
<tr>
<td>LS-5</td>
<td>13.5</td>
<td>12.5</td>
<td>623.79</td>
<td>462.01</td>
</tr>
<tr>
<td>LS-6</td>
<td>15.63</td>
<td>14.63</td>
<td>674.02</td>
<td>690.28</td>
</tr>
<tr>
<td>LS-7</td>
<td>16.34</td>
<td>15.34</td>
<td>644.74</td>
<td>584.26</td>
</tr>
<tr>
<td>LS-8</td>
<td>16.34</td>
<td>15.34</td>
<td>635.72</td>
<td>580.20</td>
</tr>
<tr>
<td>LS-9</td>
<td>16.34</td>
<td>15.34</td>
<td>693.09</td>
<td>649.19</td>
</tr>
<tr>
<td>LS-10</td>
<td>17.05</td>
<td>16.05</td>
<td>770.33</td>
<td>700.14</td>
</tr>
<tr>
<td>LS-11</td>
<td>17.05</td>
<td>16.05</td>
<td>689.93</td>
<td>551.05</td>
</tr>
<tr>
<td>LS-12</td>
<td>17.76</td>
<td>16.76</td>
<td>769.04</td>
<td>713.32</td>
</tr>
</tbody>
</table>

divide objective space into two classes for small and large sized test problems. In each class, 120 random test problems were generated according to Table 1 (i.e. 10 random samples for each test problem).

The linear statistical model can be described as follows:

\[ y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \epsilon_{ijk} \]

where \( \mu \) is the overall mean effect, \( \tau_i \) is the effect of the \( i \)-th method, \( \beta_j \) is the effect of the \( j \)-th objective space, \( (\tau\beta)_{ij} \) is the effect of interaction between the \( i \)-th method and the \( j \)-th objective space, and \( \epsilon_{ijk} \) is a random error component. The following three hypotheses were defined:

(i) For methods:

\[ H_0 : \tau_1 = \tau_2 = 0 \]
\[ H_1 : \tau_1 \neq \tau_2 \]

(ii) For objective space:

\[ H_0 : \beta_1 = \beta_2 = \cdots = \beta_{10} = 0 \]
\[ H_1 : \text{at least one } \beta_j \neq 0 \]

(iii) Interaction between methods and objective spaces:

\[ H_0 : (\tau\beta)_{ij} = 0 \]
\[ H_1 : \text{at least one } (\tau\beta)_{ij} \neq 0 \]

The results of ANOVA for each performance metric in the first and second classes have been summarized in Table 8. However, the following observations could be realized from the results of ANOVA experiments:

For small sized experiments (class 1) and N.P.S. metric: there is a significant interaction between method and objective space since the corresponding P-value is less than \( \alpha = 0.05 \). However, the main effect of methods and objective space are not significant. Although the main effect of methods is not significant, better solutions are found by the proposed MOPSO.

For small sized experiments (class 1) and diversity metric: method and objective space have a significant effect on diversity metric because the corresponding P-values are less than \( \alpha = 0.05 \). The effect of method on diversity is more than that of objective space. As the P-value for interaction is more than the level of significance, a significant interaction between method and objective space cannot be concluded. The proposed MOPSO has an advantage over the CMOPSO and promotes the greater search because of higher value for diversity metric.

For small sized experiments (class 1) and spacing metric: For this metric, the main effect of method is significant. However, the
main effect of the objective space and interaction between method and objective space are not significant. MOPSO is preferred over CMOPSO in terms of solutions spread. Moreover, its uniformity is better in comparison with CMOPSO because of lower value for spacing metric.

For large sized experiments (class 2) and N.P.S. metric: There is a significant interaction between method and objective space. In contrast to the first class, objective space has a significant effect on diversity metric. However, the main effect of methods is not significant based on N.P.S. Although the main effect of methods is not significant, better solutions are found by the MOPSO.

For large sized experiments (class 2) and diversity metric: Method and objective space have a significant effect on diversity metric. The effect of method on diversity is more than that of objective space. As the P-value for interaction is more than the level of significance, a significant interaction between method and objective space cannot be concluded. The proposed MOPSO outperforms CMOPSO and supports the larger search because of higher value for diversity metric.

For large sized experiments (class 2) and spacing metric: For this metric, the main effect of method and interaction is significant while in the first class the effect of the interaction was not significant. However, the main effect of the objective space is not significant. MOPSO is preferred over CMOPSO in terms of solutions spread. Moreover, its uniformity is better in comparison with CMOPSO because of lower value for spacing metric.

Based on our experiments, it can be concluded that the method has a dominant effect on the performance metrics. Consequently, MOPSO outperforms CMOPSO based on our experiments.

8. Conclusions

This paper proposes a new fuzzy multi-objective programming model for solving an unrelated parallel machine scheduling problem when non-zero ready times, sequence and machine dependent setup times, and secondary resource constraints are taken into account simultaneously. In order to find locally Pareto-optimal frontier, a posteriori optimization approach, i.e. a MOPSO is proposed to minimize total weighted flow time, total weighted tardiness and total machine load variation concurrently. The centroid-based distance method was used to apply dominance concept in fuzzy environment. In addition, a new selection regime for the front was proposed that integrates two common selection methods in the literature (i.e. ‘unrestricted group and ‘restricted’ group). This procedure helps maintaining diversity by pushing particles toward different parts of the Pareto front. It is allowed for each particle to have an archive to store more personal bests by applying Pareto dominance. Performance of the proposed MOPSO algorithm was compared with a conventional approach called CMOPSO. To validate the efficiency and effectiveness of the proposed algorithm, three performance metrics were used. Experimental results show that the proposed MOPSO outperforms the CMOPSO in all of these metrics.

There are some possible directions for further research. Among them, it would be nice to include some indications regarding the quality of the solutions obtained using the proposed fuzzy model in comparison with the corresponding ‘crisp’ model. Noteworthy, there are many situations where the consideration of uncertainty leads to better solutions. Many scholars show that a few of scheduling problem studies take uncertainty into account, in particular the imprecision in time estimates [78]. Nevertheless, we proposed a new MOPSO algorithm to cope with both imprecision of some critical input data and computational complexity of formulated multi-objective which demonstrated promising results for solving such scheduling problems in real sizes.

References


