Int J Adv Manuf Technol
DOI 10.1007/s00170-015-8075-1

ORIGINAL ARTICLE

Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem

Manas Ranjan Singh[1] · **Madhusmita Singh**[2] · **S. S. Mahapatra**[1] · **Nibedita Jagadev**[3]

Received: 18 August 2014 / Accepted: 3 November 2015

Abstract Due to intense competition in the market place, effective scheduling has now become an important issue for the growth and survival of manufacturing firms. To sustain in the current competitive environment, it is essential for the manufacturing firms to improve the schedule based on simultaneous optimization of performance measures such as makespan, flow time, and tardiness. The current paper presents a novel particle swarm optimization (PSO) algorithm for solving multi-objective flexible job shop scheduling problem with the goal of finding approximations of the optimal Pareto front. The Pareto-optimal solutions obtained through multi-objective particle swarm optimization (MOPSO) have been ranked by the composite scores obtained through maximum deviation theory (MDT) to avoid subjectiveness and impreciseness in the decision-making. The results are compared with non-dominated sorting genetic algorithm-II (NSGA-II) and multi-objective evolutionary algorithm (MOEA) in terms of four performance metrics. Twenty-eight benchmark instances from literature are solved by the proposed algorithm. It is observed that MOPSO outperforms NSGA-II and MOEA in four performance metrics in most of the instances.

Keywords Flexible job shop · Particle swarm optimization · MOPSO · Makespan · Flow time · Tardiness · Maximum deviation theory

✉ S. S. Mahapatra
mahapatrass2003@yahoo.com

Manas Ranjan Singh
manasranjan.singh@gmail.com

Madhusmita Singh
madhus_mita@yahoo.com

Nibedita Jagadev
queennibedita@gmail.com

[1] Department of Mechanical Engineering, National Institute of Technology Rourkela, Rourkela, India

[2] Department of Computer Science and Engineering, Swami Keshvanand Institute of Technology, Jaipur, Rajasthan, India

[3] Department of Computer Science and Engineering, ITER, Siksha 'O' Anusandhan University, Bhubaneswar, India

1 Introduction

Scheduling is a decision-making process that plays an important role both in manufacturing and service industries. It deals with the allocation of operations on machines (i.e., a sequence of operations on machines) in such a manner that some performance goals such as makespan, flow time, and tardiness can be minimized. Due to intense competition in the market place in terms of shorter product life cycles, customized products, and changing demand pattern, effective scheduling has now become an important issue for the growth and survival of manufacturing firms. To sustain in the current competitive environment, it is essential for the manufacturing firms to improve the schedules based on simultaneous optimization of performance measures such as makespan, flow time, and tardiness. Minimizing the makespan ensures maximization of the processor utilization, an important criterion from the managerial point of view. Mean flow time criterion bears significance from the operators' point of view as it minimizes maximum in-process time in the shop floor. Tardiness of a job equals to the amount of time required to complete the job beyond its due date. Tardiness is important from business perspective as tardy jobs may cause loss of customers and damage the reputation of the firm. Since all the scheduling criteria are important from business operation point of view, it is vital to optimize all the objectives simultaneously instead of a single objective.

Published online: 17 November 2015

According to shop environment, the shop scheduling problems can be classified into flow shop, flexible flow shop, job shop, and flexible job shop scheduling. A classical job shop scheduling problem (JSP) deals with a set of jobs to be processed by a set of machines. Each job is processed on machines in a given order with a given processing time, and each machine can process only one job at a time. In contrast, the flexible job shop scheduling problem (FJSP) is an extension of the classical job shop problem where operations are allowed to be processed on any among a set of available machines at a facility. In general, scheduling in flexible job shop environment is considered as NP-hard problem [1]. FJSP is considered to be more difficult to solve than classical JSP because it contains an additional constraint of assigning operations to machines at a facility. Further, FJSP problem becomes more difficult to solve if multiple criteria are simultaneously considered.

The solution strategy for multi-objective scheduling problem (MOSP) is roughly classified into two types such as weighting approach and Pareto-based approach. The weighting approach usually solves by transforming the multi-objective problem into a single-objective problem through assigning different weights for objectives. The common combination function is known as linear weighted function. However, linear weighted function might not always be able to represent the trade-off relationship between the objectives because determination of weights for objectives is a difficult task. Xia and Wu [2], Tay and Ho [3], and Li et al. [4] have proposed various algorithms to solve FJSP using weighting approach. The Pareto approach, on the other hand, provides an alternative approach for multi-objective optimization. Multi-objective flexible job shop scheduling problem (MFJSP) has been solved incorporating Pareto-optimal criteria in various algorithms like particle swarm optimization and genetic algorithm [5–11]. In this work, a multi-objective particle swarm optimization (MOPSO) technique is proposed for solving the FJSP with an objective to minimize makespan, mean flow time, and mean tardiness with the goal of finding approximations of the optimal Pareto front. In multi-objective optimization problems, convergence and diversity are two important issues. The former specifies the algorithm's capability to find the true Pareto optimal solutions and the latter imitates the algorithm's ability to find as much as possible diverse Pareto optimal solutions. In order to improve diversity, a popular operator in genetic algorithm known as mutation is embedded in the standard MOPSO algorithm to escape from local optima [12]. However, MOPSO results in a large number of non-dominated solutions. Therefore, maximum deviation theory proposed by Wang [13] has been adopted for ranking the solution to ease the decision-making process of choosing the best solution from a set of non-dominated solutions.

2 Literature review

In the past, scheduling problem in a hybrid or flexible job shop has received attention of researchers because of its importance from both theoretical and practical points of view. Brandimarte [14] was the first to apply the decomposition approach combining some existing dispatching rules for routing and Tabu search heuristic for scheduling the FJSP. Perez et al. [15] have proposed a new hierarchical heuristic algorithm for multi-objective flexible job shop scheduling problems. Xing et al. [16] have proposed a knowledge-based ant colony optimization (KBACO) algorithm for solving the FJSP. Bagheri et al. [17] have employed an artificial immune algorithm to solve the flexible job shop problem. Chang et al. [18] have proposed the gradual priority weighting approach to search the Pareto optimal solution for multi-objective FJSP. The approach rests on searching the feasible solution space for the first objective at the beginning, and the search proceeds towards other objectives step by step. Further, the multi-objective scheduling problems consider makespan, total flow time, total tardiness, and maximum tardiness as the performance measures.

Suresh and Mohanasundaram [19] have applied Pareto archived simulated annealing to the multi-objective job shop scheduling problem in which the related objectives are minimization of makespan and mean of flow time. Kacem et al. [5] have proposed a localization approach to solve the resource assignment problem and an evolutionary approach controlled by the assignment model to solve the mono-objective and multi-objective FJSP. Coello et al. [20] have proposed an approach in which Pareto dominance is combined into particle swarm optimization in order to allow the heuristic to handle problems with several objective functions. Lei [21] presents a particle swarm optimization for multi-objective job shop scheduling problem in order to simultaneously minimize makespan and total tardiness of jobs. Lei and Wu [22] have proposed a crowding measure-based multi-objective evolutionary algorithm (CMOEA) which makes use of the crowding measure to adjust the external population and assign different fitness for individuals. The comparison between CMOEA and strength Pareto evolutionary algorithm (SPEA) reveals that CMOEA performs better in job shop scheduling with two objectives like minimization of makespan and total tardiness. Li et al. [9] have presented a novel discrete artificial bee colony (DABC) algorithm for solving the multi-objective flexible job shop scheduling problem with maintenance activities. The considered performance criteria are the maximum completion time, the so-called makespan, the total workload of machines, and the workload of the critical machine. Moslehi and Mahnam [8] have suggested a Pareto approach hybridizing particle swarm algorithm and local search to solve multi-objective FJSP. However, managing external archive and selection of best solution from a set of Pareto solutions is difficult in multi-objective optimization. In this paper, the search mechanism of the particle

swarm optimization is considered due its effectiveness to solve different objectives simultaneously in FJSP addressing external archive management and selection of best solution using maximum deviation theory.

3 Flexible job shop scheduling

The FJSP is considered as an extension of the traditional job shop scheduling problem with added constraint of an operation of a job can be processed in more than one facility. The problem is treated as non-deterministic polynomial-time hard (NP-hard). The critical issues to solve the problem are as follows: (1) assignment of operations to alternative machines, (2) sequencing the operations in each machine, and (3) rescheduling policy when a disruption occurs. The several assumptions and constraints in a hypothetical or deterministic FJSP are usually formulated as follows:

The flexible job shop problem is to organize the execution of n jobs on m machines. In this problem, there are a set of machines, $k=1,2,\ldots,m$, and a set of jobs, $i=1,2,\ldots,n$, so that each job consists of a predetermined sequence of operations. Each operation requires one machine out of a set of available machines. All jobs and machines are available at time zero and a machine can only execute one operation at a given time. Preemption is not allowed, i.e., each operation must be completed without interruption once it starts. The FJSP is machine dependent because the performance of each operation on each allowable machine has a different processing time. The objective of the problem is to assign each operation to an appropriate machine and sequence the operations on the machines in order to minimize the makespan which is the time required completing all the jobs.

3.1 Problem representation

In this work, a real number encoding system is proposed. The integer part is used to assign the operations of each job to the machine, and fractional part is used to sequence of the operations on each machine. The position of each particle is represented by a real number. The value of integer part allocates as a priority level for each operation which is used to select the machine for the operation. First sequencing of available machines for an operation according to the increasing order of processing time is carried out. If tie occurs, the machine having lower index number is given the priority. Priority levels for all machines are generated for processing all the operations of each job [2]. As an instance, a problem is to execute three jobs on four machines. Table 1 represents data including jobs, operations, and processing times on different machines. Table 2 shows the order of priority or priority level, i.e., 1, 2, 3, and 4 of machines corresponding to each operation.

Table 1 Example problem

Job	Operations	Machine 1	Machine 2	Machine 3	Machine 4
Job 1	$O_{1,1}$	9	5	4	3
	$O_{1,2}$	7	8	9	5
	$O_{1,3}$	5	8	8	3
Job 2	$O_{2,1}$	4	6	5	8
	$O_{2,2}$	5	4	6	2
Job 3	$O_{3,1}$	3	8	6	3
	$O_{3,2}$	5	5	2	2

Table 3 represents the stochastic particle position representation. Initial particle positions in the swarm are generated by random number distributed uniformly on $[x_{min}, x_{max}]$ where $x_{min}=1.0$, $x_{max}=$mpl. The maximum position x_{max} of the particle is taken as the maximum value of priority level (mpl), i.e., the number of machines available. The position of the particle must be a positive integer as each particle position value represents priority level for each operation. Hence, it lies in the range [1, mpl]. For example, the 1st position is 2.25 and the integer value is 2. Therefore, operation $O_{1,\ 1}$ is assigned to machine 3 as per the priority order in Table 2. The process order of operations to be scheduled on the same machine depends on the value of fractional parts. The operations are sequenced according to the ascending order of the fractional part which is processed by the same machine. For instance, operations $O_{1,2}$ and $O_{3,2}$ are assigned to machine 2. The sequence of operations to be scheduled on machine 2 is operation ($O_{3,2}$) followed by operation ($O_{1,2}$) because the fractional part of the particle position for $O_{3,2}$ is greater than fractional part of the particle position for $O_{1,2}$. If the value of fractional parts is equal then the operation processing sequence is randomly chosen.

4 Multi-objective optimization

Multi-objective optimization is defined as the problem of finding a vector of decision variables that satisfies all constraints and simultaneously optimizes a vector function whose

Table 2 Priority order

Job	Operations	Priority 1	Priority 2	Priority 3	Priority 4
Job 1	$O_{1,1}$	M4	M3	M2	M1
	$O_{1,2}$	M4	M1	M2	M3
	$O_{1,3}$	M4	M1	M2	M3
Job 2	$O_{2,1}$	M1	M3	M2	M4
	$O_{2,2}$	M4	M2	M1	M3
Job 3	$O_{3,1}$	M1	M4	M3	M2
	$O_{3,2}$	M3	M4	M1	M2

Table 3 A stochastic particle position representation

Operation	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{2,1}$	$O_{2,2}$	$O_{3,1}$	$O_{3,2}$
Particle positions	2.25	3.64	1.12	2.44	3.14	2.05	4.82
Priority level	2	3	1	2	3	2	4
Processing machine	M3	M2	M4	M3	M1	M4	M2

elements represent the objective functions. Mathematically, the multi-objective optimization (MOO) problem can be formulized as follows:

$$\text{Minimizing (or Maximizing)}\, F(x) = \left\{ f_1(x), f_2(x), \ldots . f_q(x) \right\} \quad \text{subject to} \quad g(x) \leq 0, \quad h(x) = 0$$

A MOO solutions minimizes (or maximizes) the components of a vector $F(x)$ where x is n-dimensional decision variable vector $X=(x_1,x_2,\ldots\ldots\ldots x_n)$ and $g(x) \leq 0, \quad h(x) = 0$ are set of constraints that determine the feasible solution area in minimizing (or maximizing) $F(x)$ with 'q' objective functions. In this study, the following objectives of FJSP are to be minimized as follows:

Objective 1 (F_1): The first objective is to minimize the makespan ($C_{\max}$), i.e., the completion time of all jobs in the last stage. $C_{\max}=\max\{C_i\}$ where C_i is the completion time of job i at last stage

Objective 2 (F_2): The second objective is to minimize the mean tardiness $\left(\overline{T}\right)$, i.e., the amount of time by which the completion time of job i differs from the due date.

$$\overline{T} = \frac{1}{n} \sum_{i=1}^{n} \max\{0, (C_i - d_i)\} \quad \max\{0, (C_i - d_i)\} \quad (1)$$

where d_i is the due date of job i

Objective 3 (F_3): The third objective is to minimize the mean flow time $\left(\overline{F}\right)$, i.e., the amount of time spent by job in the shop.

$$\overline{F} = \frac{1}{n} \sum_{i=1}^{n} (C_i - r_i) \quad (2)$$

where r_i is the release date.

5 Particle swarm optimization

Particle swarm optimization (PSO) algorithm, originally introduced by Eberhart and Kennedy [23], is a population based evolutionary computation technique motivated by the behavior of organisms such as bird flocking and fish schooling. In PSO, each member is called particle and each particle moves around in the search space with a velocity which is continuously updated by the particle's individual contribution and the contribution of the particle's neighbors or the contribution of the whole swarm. The members of the whole population are maintained during the search procedure so that information is socially shared among all individuals to direct the search towards the best position in the search space. Each particle moves towards its best previous position and towards the best particle in the whole swarm called the g_{best} based on the global neighborhood. Each particle moves towards its best previous position and towards the best particle in its restricted neighborhood based on the local variant so-called the p_{best} model. PSO is basically characterized as a simple heuristic of well-balanced mechanism with flexibility to progress and adjust to both global and local exploration capabilities. All the particles tend to converge to the best solution rapidly even in the local version in most cases as compared to genetic algorithm. Due to the simple concept, easy implementation, and rapid convergence, PSO has gained much attention and been successfully applied to a wide range of applications such as job scheduling, power and voltage control, mass spring system, supply chain network, and vehicle routing problems [12, 24–28].

In PSO, the initial population is generated randomly and parameters are initialized. After evaluation of the fitness function, the PSO algorithm repeats the following steps iteratively:

- Personal best (best value of each individual so far) is updated if a better value is discovered.
- Then, the velocities of all the particles are updated based on the experiences of personal best and the global best in order to update the position of each particle with the velocities currently updated.

After finding the personal best and global best values, velocities and positions of each particle are updated using Eqs. 3 and 4, respectively.

$$v_{ij}^{t} = w^{t-1} v_{ij}^{t-1} + c_1 r_1 \left(p_{ij}^{t-1} - x_{ij}^{t-1} \right) + c_2 r_2 \left(g_{ij}^{t-1} - x_{ij}^{t-1} \right) \quad (3)$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \quad (4)$$

where v_{ij}^{t} represents velocity of particle i at iteration t with respect to j^{th} dimension (j=1,2,……n). p_{ij}^{t} represents the position value of the i^{th} personal best with respect to the j^{th} dimension. g_{ij}^{t} represents the global best (g_{best}), i.e., the best of p_{best} among all the particles. x_{ij}^{t} is the position value of the i^{th} particle with respect to j^{th} dimension. c_1 and c_2 are positive acceleration parameters which provide the correct balance between exploration and exploitation and are called the cognitive parameter and the social parameter, respectively. r_1 and r_2 are the random numbers provide a stochastic characteristic for the particles velocities in order to simulate the real behavior of the birds in a flock. The inertia weight parameter w is a control

parameter which is used to control the impact of the previous velocities on the current velocity of each particle. Hence, the parameter w regulates the trade-off between global and local exploration ability of the swarm. The recommended value of the inertia weight w is to set it to a large value for the initial stages in order to enhance the global search of the search space and gradually decrease it to get more refined solutions facilitating the local search in the last stages. In general, the inertia weight is set according to the following Eq. 5 (Modares et al. 2011).

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter} \tag{5}$$

where $w_{\min}$ and $w_{\max}$ are initial and final weights and $\text{iter}_{\max}$ is the maximum number of iterations and iter is the current iteration number.

6 Multi-objective particle swarm optimization (MOPSO)

Multi-objective optimization (MOO) has been an active area of research in last two decades. Such problems arise in many applications where two or more objective functions have to be optimized simultaneously. PSO has been extended for solving the MOO problems, which is generally known as the multi-objective particle swarm optimization (MOPSO). The main difference between a basic PSO (single-objective) and MOPSO is the distribution of g_{best}. In MOPSO algorithm, g_{best} must be redefined in order to obtain a set of non-dominated solutions (Pareto front). In single-objective problems, there is only one g_{best} exists. In MOO problems, more than one conflicting objectives will be optimized simultaneously. There are multiple numbers of non-dominated solutions which are located on or near the Pareto front. Therefore, each non-dominated solution can be the g_{best}. Extending PSO to handle multi-objectives has been proposed by Mostaghim and Teich [29] and Wang and Singh [30]. Coello et al. [20] have proposed a MOPSO algorithm which adopts an external repository and mutation operator for finding out Pareto-optimal set of solutions.

6.1 Proposed MOPSO algorithm

Real world problems involve simultaneous optimization of numerous contradistinctive and conflicting nature objectives. When all objectives are considered, these solutions are optimum in the sense that none of the other solutions in the search area are exceptionally good to another solution. These solutions are called as Pareto-optimal solutions. The image of the efficient set in the objective space is named as non-dominated set as each solution dominates the other solution. To identify the non-dominance, each solution is compared with every single solution and checked for satisfying the rules given below for the solution under consideration.

$$\text{Obj.1[l]} > Obj.1\text{[m]} \text{ and } \text{Obj.2[l]} \geq \text{Obj.2[m]} \tag{6}$$

$$\text{Obj.1[l]} \geq \text{Obj.1[m]} \text{ and } \text{Obj.2[l]} > Obj.2\text{[m]} \tag{7}$$

where l and m correspond to solution number in the population. Obj.1 and Obj.2 are two objective function values.

The multi-objective optimization aims at two objectives:

(a) Converging to the Pareto-optimal solution set;
(b) Maintaining diversity and distribution in solutions.

While solving single-objective optimization problems, the g_{best} that each particle uses to update its position is completely determined once a neighborhood topology is established. However, in the case of multi-objective optimizations problems, each particle might have a set of g_{best} from which just one can be selected in order to update its position. Such set of g_{best} is usually stored in a different place from the swarm known as external archive 'A_t'. This is a repository in which the non-dominated solutions found so far are stored. The MOPSO maintains an external archive 'A_t'of non-dominated solutions of the population which is updated after every iteration. The global archive 'A_t'is empty in the beginning and can store a user-specified maximum number of non-dominated solutions. In case the number of non-dominated solutions exceeds the maximum size of the archive, some individuals are cropped. There are several methods of controlling the external archive such as Maximin fitness based size control [31], epsilon-dominance based size control [29], and crowding distance based size control [32]. Archive size control is critical because the number of non-dominated solutions can grow very fast although there are studies where archive size is unconstrained [33].

Crowding distance technique has been extensively applied in evolutionary multi-objective algorithms to promote diversity. The use of crowding distance measure in MOPSO for g_{best} selection was first made in Raquel and Naval [32]. The approach is quite capable in converging towards the Pareto front and generating a well-distributed set of non-dominated solutions. In this study, crowding distance approach has only been applied to make g_{best} selection. Crowding distance factor is defined to show how much a non-dominated solution is crowded with other solutions. The crowding distance (CD) factor of a solution provides an estimate of the density of solutions surrounding that solution [34, 35]. Figure 1 shows the calculation of the crowding distance

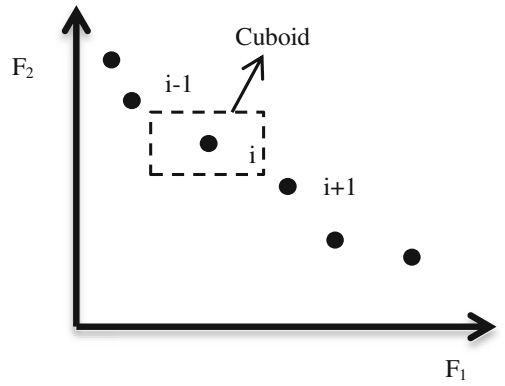

Fig. 1 The crowding distance

of point k which is an estimate of the size of the largest cuboid enclosing k without including any other point. CD factor of boundary solutions which have the lowest and highest objective function values ($f_{\max}$ and $f_{\min}$, respectively) are given an infinite crowding distance values. For other solutions, CD factor for the solution k is calculated by following relation.

$$\mathrm{CD}_k = \frac{\left(f_{k+1} - f_{k-1}\right)}{\left(f_{\max} - f_{\min}\right)} \tag{8}$$

Finally, the overall crowding factor is computed by adding the entire individual crowding distance values in each objective function.

The non-dominated solutions in 'A_t'are sorted in descending crowding distance values and top 10 % of them are randomly used as g_{best} guides.

Particle swarm optimization typically converges relatively rapidly at the beginning of the search and then slows down or stagnates due to loss of diversity in the population [12, 36]. To overcome this drawback, mutation, a widely used operator in genetic algorithm, is used to introduce diversity in the search procedure. When the change of the whole archive tends to decrease, the mutation process will begin. If the number of iteration is less than the product of maximum number of iteration and probability of mutation then only the mutation is performed on the position of the particle. Given a particle, a randomly chosen variable, say m_p, is mutated to assume a value m_p' as given by following equation.

$$m_p' = \begin{cases} m_p + \Delta\left(t, \mathrm{UB} - m_p\right) & \text{if flip} = 0 \\ m_p - \Delta\left(t, m_p - \mathrm{LB}\right) & \text{if flip} = 1 \end{cases} \tag{9}$$

when flip denotes the random event of returning 0 or 1. UB and LB denote the upper and lower bound of the variable m_p, respectively. The function $\Delta(t,x)$ returns a value in the range $[0,x]$ such that the probability of $\Delta(t,x)$ being close to 0 increases as t increases.

$$\Delta(t,x) = x \times \left(1 - r^{\left(1 - \frac{t}{\mathrm{MAXT}}\right)^b}\right) \tag{10}$$

where r is the random number generated in the range [0, 1], MAXT is the maximum number of iterations and t is the number of iteration. The parameter b determines the degree of dependence of mutation on the iteration number.

To summarize, the main difference between a basic PSO (single-objective) and MOPSO is the distribution of g_{best}. In single-objective problems, there is only one g_{best} exists. In MOPSO algorithm, g_{best} must be redefined in order to obtain a set of non-dominated solutions (Pareto front). Therefore, multiple numbers of non-dominated solutions are located on or near the Pareto front. Each non-dominated solution can be a g_{best}. The important feature of MOPSO is that the individuals also maintain a personal archive which is known as p_{best} archive with a maximum size. The p_{best} archive contains the most recent non-dominated positions a particle has encountered in the past. In every iteration t, each particle i is allocated with two guides p_{best} and g_{best} from its p_{best} archive and swarms global archive 'A_t'. After the guide selection, positions and velocities of particles are updated according to the Eqs. 11 and 12 where v_{ij}^t represents velocity and x_{ij}^t is the position value of the i^{th} particle with respect to j^{th} dimension. Maximum number of generations is set as termination criterion. The complete algorithm for MOPSO is shown as follows:

6.2 MOPSO algorithm

1. For i=1 to M (M is the population size)
   a. Initialize position of the particles randomly
   b. Initialize v_{ij}^t=0 (v is the velocity of each particle)
   c. Evaluate each particle's fitness
   d. Compare each particle's fitness with the particle's p_{best}. *Compare the fitness with the population's overall previous best*
   e. Find out the personal best (p_{best}) and global best (g_{best}).
2. End For
3. Initialize the iteration counter t=0
4. Store the non-dominated vectors found into archive 'A_t' ('A_t'is the external archive that stores non-dominated solutions found)
5. Repeat
   a. Compute the crowding distance values of each non-dominated solution in the archive 'A_t'

b. Sort the non-dominated solutions in 'A_t'in descending crowding distance values
c. For i=1 to M

i. Randomly select the global best guide from a specified top 10 % of the sorted archive 'A_t' and store its position to g_{best}.
ii. Compute the new velocity:

$$v_{ij}^{t} = w^{t-1}v_{ij}^{t-1} + c_1 r_1\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2\left((A_t)_{ij}^{t-1} - x_{ij}^{t-1}\right) \quad (11)$$

($(A_t)_{ij}^{t-1}$ is the global best guide for each non-dominated solution)
iii. Calculate the new position of

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \quad (12)$$

iv. If (t<(MAXT*PMUT), then perform mutation on x_{ij}^{t}.
(MAXT is the maximum number of iterations and PMUT is the probability of mutation)
v. Evaluate x_{ij}^{t}

d. End For
e. Insert all new non-dominated solution into archive 'A_t' if they are not dominated by any of the stored solutions. All dominated solutions in the archive are removed by the new solution from the archive. If the archive is reached its maximum, the solution to be substituted is determined by the following steps:

i. Compute the crowding distance values of each non-dominated solution in the archive 'A_t'
ii. Sort the non-dominated solutions in archive 'A_t'in descending crowding distance values
iii. Randomly select a particle from a specified bottom 10 % of the sorted archive 'A_t' and replace it with the new solution

f. Increment iteration counter t
g. Update the personal best solution of each particle. If the current p_{best} dominates the position in the memory, the particle position is updated.

6. Until maximum number of iterations is reached

6.3 Solution ranking by maximum deviation theory

Since MOPSO results in a large number of non-dominated solutions, choosing a best solution depends on decision-maker's judgement and intuition. Usually, multi-attribute decision-making (MADM) approaches are adopted to obtain scores for the solutions and the solution exhibiting maximum score is selected as the best one. However, the weights assigned in multi-attribute decision-making process for converting multiple objectives into a single equivalent objective score are reasonably subjective in nature and affect the decision of ranking the alternative solutions considerably. In order to avoid uncertainty of subjective assigning of weights from the experts and extract the accurate information from the available data, maximum deviation theory (MDT) suggested by Wang [13] is adopted in this work. The basic idea of MDT rests on smaller weight should be assigned to the attribute having similar values in comparison to the attribute having larger deviations.

The non-dominated solutions obtained in MOPSO solutions are used as the decision matrix. Every element of the decision matrix denotes the value of j^{th} attribute for i^{th} alternative where i=1, 2 …n, and j=1, 2…m. Normalization of each attribute is carried out to transform different scales and units among various attributes into a common measurable scale. The normalization of the attribute depends on its type such as "higher the better" and "lower the better." The following equations are used for normalization of attributes.

$$x_{ij}^{*} = \frac{\max_i\{x_{ij}\} - x_{ij}}{\max_i\{x_{ij}\} + \min_i\{x_{ij}\}}, \quad \text{for lower the better attributes} \quad (13)$$

$$x_{ij}^{*} = \frac{x_{ij} - \min_i\{x_{ij}\}}{\max_i\{x_{ij}\} - \min_i\{x_{ij}\}}, \quad \text{for higher the better attributes} \quad (14)$$

The difference of performance values for each alternative is computed. For the attribute $\{A_j|\ j=1, 2\ldots m\}$, the deviation value of the alternative $\{S_i|\ i=1, 2\ \ldots.n\}$ from all the other alternatives can be computed by the following equation.

$$D_{ij}(w_j) = \sum_{i=1}^{n} d\left(\tilde{r}_{ij}, \tilde{r}_{lj}\right) w_j \quad (15)$$

where w_j is the weight of the attributes to be calculated and $D_{ij}(w_j)$ is the deviation value of the alternatives.

The total deviation values of all alternatives with respect to other alternatives for the attribute $\{A_j|\ j=1, 2\ldots m\}$ can be computed by the following relation.

$$D_j(w_j) = \sum_{i=1}^{n} D_{ij}(w_j) = \sum_{i=1}^{n}\sum_{i=1}^{n} d\left(\tilde{r}_{ij}, \tilde{r}_{lj}\right) w_j \quad (16)$$

where $D_j(w_j)$ is the total deviation value of all the alternatives.

The deviation of all the attributes along all the alternatives can be calculated by the relation.

$$D(w_j) = \sum_{j=1}^{M} D_j(w_j) = \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij}, \tilde{r}_{lj}\right) w_j \quad (17)$$

where $D(w_j)$ is deviation of all the attributes along all the alternatives.

A linear programming model is constructed for finding out the weight vector w to maximize all deviation values for all the attributes and is given by as follows:

$$\begin{cases} D(w_j) = \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{lj}\right) w_j \\ \text{s.t}\sum_{j=1}^{M} w_j^2 = 1,\ w_j \geq 0, j = 1,2,\ldots\ldots,M \end{cases} \tag{18}$$

A Lagrange function is constructed for solving the above model.

$$L(w_j,\alpha) = \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{lj}\right) w_j + \alpha\left(\sum_{j=1}^{M} w_j^2 - 1\right) \tag{19}$$

where α is the Lagrange multiplier. The partial derivative of L (w_j,α) with respect to w_j and α are as follows:

$$\begin{cases} \frac{\partial L}{dw_j} = \sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{lj}\right) + 2\propto w_j = 0 \\ \frac{\partial L}{d\propto} = \sum_{j=1}^{M} w_j^2 - 1 = 0 \end{cases} \tag{20}$$

Further, w_j and α values are calculated from Eqs. 19 and 20

$$\begin{cases} 2\propto = -\sqrt{\sum_{j=1}^{M}\left(\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{ij}\right)\right)^2} \\ w_j = \dfrac{\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{ij}\right)}{\sqrt{\sum_{j=1}^{M}\left(\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{ij}\right)\right)^2}} \end{cases} \tag{21}$$

The normalized attribute weights can be further determined by the following relation.

$$w_j = \frac{\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{ij}\right)}{\sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{l=1}^{N} d\left(\tilde{r}_{ij},\tilde{r}_{ij}\right)} \tag{22}$$

The non-dominated solutions obtained through MOPSO algorithm are ranked by estimating the composite score of each solution by addition of the weighted performance of all attributes. Considering the ranking of the solutions, the scheduler may choose suitable parametric setting from the top ranking solutions to justify the objectives set by the industry.

7 Result and discussion

In the present work, multi-objective particle swarm optimization (MOPSO) has been developed for solving the flexible job shop scheduling problem (FJSP) with bi-objective criteria, i.e., minimize makespan as primary objective and mean flow time and mean tardiness as secondary objective with the goal of finding approximations of the optimal Pareto front. In the proposed MOPSO algorithm, problem representation presented in section 3 is used to solve the FJSP. The algorithm is implemented in Matlab 7 on a Pentium IV running at 2 GHz on the Windows XP operating system. Simulation study is carried out to demonstrate the potentiality of MOPSO algorithm. A trial run is carried out for a set of problems to set the algorithm parameters. The initial population chosen for the algorithms is 80. The parameters employed for MOPSO are as follows: the size of archive is 100, the inertia weight is 0.4, and both the cognitive and social parameters (c1 and c2) are taken as 2. A maximum CPU time limit of 1000 s is set for each run of the algorithm to ensure that all algorithms are executed for the same CPU time for a meaningful comparison.

The proposed algorithm is tested on two sets of problem instances from Brandimarte [14] and Dauzere-peres [37] (DP data). Brandimarte [14]'s (BR) data set contains a set of 10 problems denoted as Mk01 to Mk10. The number of jobs ranges from 10 to 20, the number of machine ranges from 4 to 15, and the number of operations for each job ranges from 5 to 15. These two data sets are the most commonly adopted benchmark instances in the literature on FJSP. The DP data set is a set of 18 problems referred as 1a to 18a. The number of jobs ranges from 10 to 20, the number of machine ranges from 5 to 10, and the number of operations for each job ranges from 15 to 25. The effectiveness of the proposed MOPSO algorithm is compared with two popular multi-objective algorithm known as multi-objective evolutionary algorithm (MOEA) and non-dominated sorting genetic algorithm II (NSGA-II) which is successfully applied in many multi-objective problems [27, 38–40].

In Pareto approach, the solutions are compared based on the Pareto dominance relation. Solution 'A' dominates solution 'B', if 'A' is not worse than 'B' for all objectives or is better than 'B' for at least one objective. Solution 'A' is Pareto optimal if it is not dominated by any other solution. The Pareto approach produces a set of Pareto optimal solutions which represent the trade-off between objectives through the distribution of obtained solutions. Pareto front determined by evaluating each member of the Pareto optimal solution set. The user can select the favorite solution directly from the number of Pareto optimal solutions.

Based on exhaustive experimentation, Figs. 2 and 3 are drawn to show the Pareto front between makespan and mean flow time and makespan and mean tardiness for the benchmark instances of FJSP. The Pareto fronts reveal that a small decrease of makespan can cause a large increase in the other

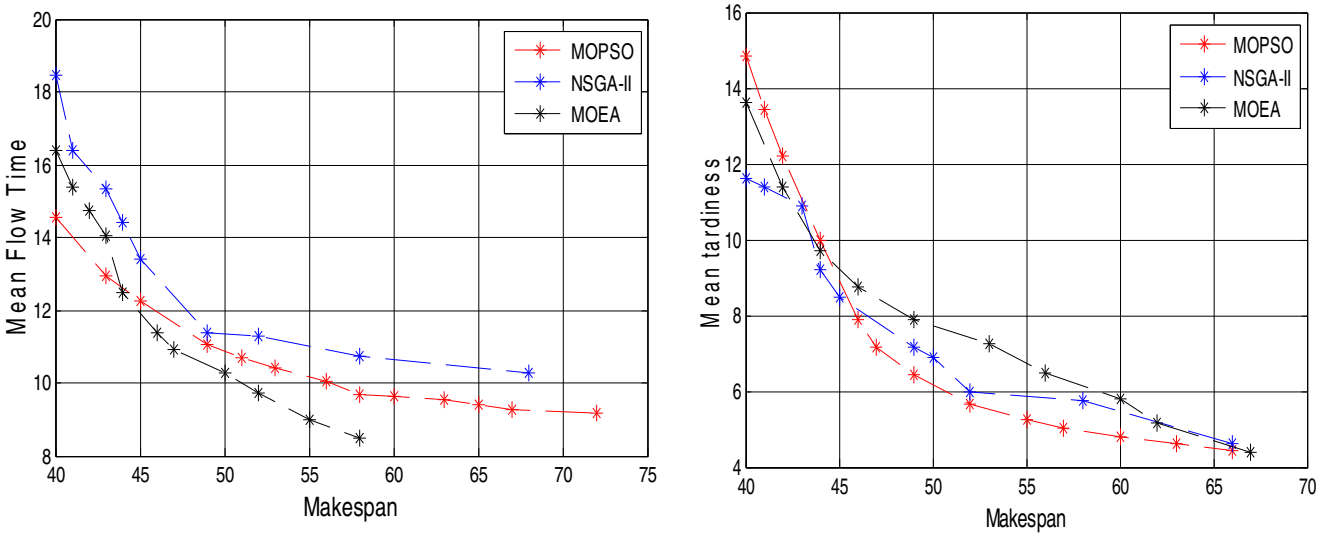

Fig. 2 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance Mk01

conflicting objective. The results convey two messages: (1) Focusing on optimizing a single objective may result in bad performance of the other objective and (2) The trade-off relationship between the objectives is not always easy to predict.

7.1 Performance measures

There are two goals in a multi-objective optimization: (1) convergence to the Pareto-optimal set and (2) maintenance of diversity in solutions of the Pareto-optimal set. These two tasks cannot be measured adequately with one performance metric. Many performance metrics have been suggested to evaluate the non-dominated solutions [34, 41]. To evaluate comprehensively the non-dominated solutions obtained by the MOPSO, NSGA-II, and MOEA algorithm, four performance metrics are adopted in this paper. The following performance measures are used to compare the results of non-dominated solutions obtained by multi-objective algorithms.

Mean ideal distance (MID) The MID measurement presents the proximity between non-dominated solutions and ideal point (0, 0). Algorithm A is considered to have more opportunity to reach the Pareto frontier than algorithm B if A has the lower value of MID than B. MID of algorithm can be obtained by the following formulation.

$$\mathrm{MID} = \frac{\sum_{i=1}^{n} C_i}{n} \tag{23}$$

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

f_{1i} and f_{2i} are the objective function values for solution i. The performance of the algorithm will be better if the value of MID is lower.

The rate of achievement to two objectives simultaneously (RAS) The value of this measure is calculated from the

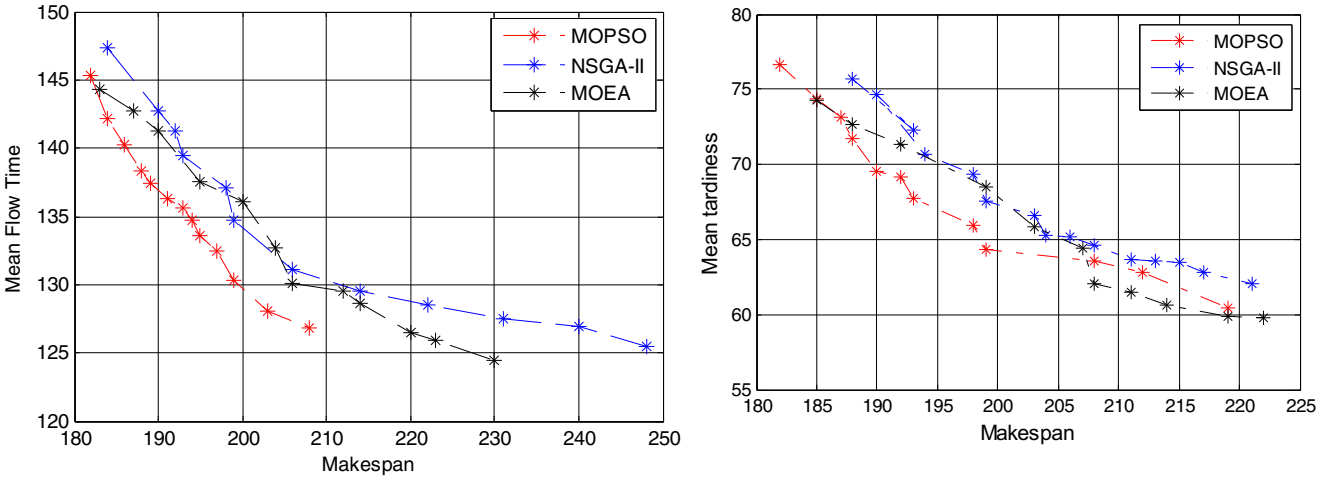

Fig. 3 Pareto front obtained by the proposed MOPSO and NSGA-II for the instance Mk05

Table 4 Performance metrics of Pareto front obtained by the objective of makespan and mean flow time

Problem	$n \times m$	MID			RAS			SNS			Diversity (DM)		
		MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA
Mk 01	10×6	*56.119*	50.93	57.226	*10.71*	12.137	12.625	*10.57*	8.269	9.6335	*32.44*	29.17	30.422
Mk 02	10×6	*43.805*	46.625	44.0402	8.977	*8.471*	9.12067	*3.926*	3.688	3.348	*17.88*	17.1918	17.73627
Mk 03	15×8	*257.273*	288.285	265.4343	*40.51*	62.709	59.5352	19.381	25.324	*26.05*	87.531	*108.847*	92.80752
Mk 04	15×8	104.0892	103.9267	*103.4669*	22.0708	*21.352*	21.6403	*10.0842*	13.052	11.547	37.824	44.8517	*45.05193*
Mk 05	15×4	236.83	249.59	*234.7948*	*18.081*	35.55	29.3417	*16.78*	14.53	15.169	30.27	*67.61*	58.54179
Mk 06	10×15	*151.58*	155.12	151.7338	*15.21*	19.37	19.1376	*2.266*	1.46	2.0707	*23.34*	21.34	22.19987
Mk 07	20×5	*222.77*	229.75	225.195	*38.49*	43.45	45.5776	*18.605*	13.37	15.579	*74.92*	67.73	70.27355
Mk 08	20×10	*644.825*	645.094	687.817	36.101	35.988	*33.295*	*6.9437*	5.5701	5.9068	*49.8*	48.2801	46.85273
Mk 09	20×10	*752.87*	755.23	753.028	*66.75*	66.87	68.4135	*23.64*	22.78	22.282	36.67	*37.12*	36.99229
Mk 10	20×15	1677.72	*1672.1*	1675.693	*347.56*	354.3	351.717	*122.45*	122.1	121.48	267.69	264.22	*276.5406*
1a	10×5	*3214.13*	3244.062	3227.937	85.71	**57**	64.1951	31.45	31.001	*32.037*	*129.76*	122.054	123.2464
2a	10×5	*3097.135*	3157.46	3099.661	371.75	*266.908*	318.189	*90.293*	58.711	67.907	*658.67*	428.78	495.4058
3a	10×5	*3494.793*	3620.75	3612.129	*836.5*	897.275	872.081	*419.971*	308.3	401.17	*2194.8*	1571.38	1982.205
4a	10×5	3141.27	3095.47	*3087.069*	207.99	236.56	*191.21*	*79.81*	48.47	62.24	*412.36*	225.51	347.4433
5a	10×5	3027.77	*3015.9*	3016.116	*214.904*	262.67	234.007	30.563	*36.16*	24.557	389.8	406.253	*423.4029*
6a	10×5	*2951.502*	3041.99	3061.718	*372.48*	429.75	403.114	*185.605*	141.98	171.49	626.06	*825.412*	719.5618
7a	15×8	3023.466	3038.891	*2029.317*	139.01	*87.5*	92.849	*29.588*	22.894	25.118	*220.99*	175.513	198.232
8a	15×8	*3057.918*	3134.15	3084.728	*471.034*	573.98	517.311	*265.033*	215.541	247.07	1362	1047.23	*1423.722*
9a	15×8	*3061.157*	3176.37	3214.41	*469.39*	492.52	473.325	*202.92*	201.47	201.74	*1068.2*	967.12	960.0263
10a	15×8	*3031.89*	3081.275	3057.498	*156.47*	167.6	169.472	35.25	*67.8*	62.416	*355.9*	322.939	341.1053
11a	15×8	*2841.59*	2908.101	2879.154	359.58	*347.66*	349.17	66.8	*97.25*	71.221	*637.96*	584.041	584.2331
12a	15×8	*3098.43*	3154.67	3152.858	345.21	346.79	*342.698*	172.34	175.22	*177.14*	*867.79*	867.45	849.7696
13a	20×10	*3329.32*	3364.32	3371.724	*262.27*	311.29	297.339	*74.311*	57.81	68.702	385.31	308.62	*391.5339*
14a	20×10	*3619.43*	3846.77	3882.401	*525.37*	556.65	548.527	*176.97*	154.88	159.72	*924.13*	894.68	907.462
15a	20×10	3087.86	3197.5	*3056.517*	*367.56*	413.46	421.791	*286.48*	267.72	280.12	*346.76*	335.41	340.6731
16a	20×10	*3438.78*	3475.25	3470.563	*481.38*	490.11	482.145	*312.98*	284.77	301.02	*189.4*	175.5	188.4641
17a	20×10	3310.2	3346.841	3342.654	*679.53*	678.87	678.274	148.83	*164.7*	155.51	*415.62*	417.23	374
18a	20×10	3864.58	3749.15	3749.977	784.29	*776.71*	779.491	*229.46*	227.69	228.18	523.84	521.92	*524.25*

The best obtained values are marked in italic number

following relation. Smaller value of this criterion indicates a higher quality solution.

$$\mathrm{RAS} = \frac{\sum_{i=1}^{n} \left|f_{1i} - f_1^{\mathrm{best}}\right| + \left|f_{2i} - f_2^{\mathrm{best}}\right|}{n} \tag{24}$$

f_1^{best} and f_2^{best} are the best solutions in the non-dominated sets for objectives 1 and 2.

Spread of non-dominance solutions (SNS) The spacing metric aims at assessing the spread (distribution) of vectors throughout the set of non-dominated solutions. This criterion, which is known as an indicator of diversity, is calculated from the following relation:

$$\mathrm{SNS} = \sqrt{\frac{\sum_{i=1}^{n} (\mathrm{MID} - C_i)^2}{n-1}} \tag{25}$$

Diversification matrix (DM) This performance measure gives an indication of the diversity of solutions obtained from a given algorithm.

$$\mathrm{DM} = \sqrt{(\max f_1 - \min f_1)^2 + (\max f_2 - \min f_2)^2} \tag{26}$$

where $\max f_1$ and $\max f_2$ is the maximum objective functions value of the non-dominated solutions and $\min f_1$ and $\min f_2$ is the minimum objective functions value of the non-dominated solutions. Larger values of SNS and DM are indicative of higher quality solutions.

The effectiveness of the algorithms is tested by solving 28 different benchmark problems of Brandimarte and Dauzere-peres data set. The results obtained by the proposed algorithms are compared in terms of the performance metrics with the NSGA-II and MOEA. Tables 4 and 5 illustrate the comparative results of two algorithms with respect to four performance measures. From Tables 4 and 5, it can be concluded

Table 5 Performance metrics results of Pareto front obtained by the objective of makespan and mean tardiness

Problem	$n \times m$	MID			RAS			SNS			Diversity (DM)		
		MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA	MOPSO	NSGA-II	MOEA
Mk 01	10×6	*48.26*	49.6	49.671	11.64	*10.37*	10.585	*7.42*	7.4063	7.1413	*27.99*	26.91	26.547
Mk 02	10×6	*34.1*	43.59	40.536	*5.15*	8.741	7.4559	82.55	*84.37*	82.189	28.65	21.013	*29.804*
Mk 03	15×8	*234.9*	259.68	254.91	*32.0726*	37.66	32.966	*16.486*	15.52	15.976	52.0522	*59.55*	55.354
Mk 04	15×8	*85.46*	108.75	86.011	*11.79*	16.6	15.104	*11.62*	6.17	8.824	*27.982*	12.6	19.113
Mk 05	15×4	207.8	217.201	*205.99*	*21.96*	24.64	22.51	9.406	10.23	*11.351*	44.391	43.75	*44.644*
Mk 06	10×15	*128.96*	131.97	129.97	6.0175	*5.979*	5.1123	*2.153*	1.274	1.243	6.562	*10.598*	6.0462
Mk 07	20×5	*193.196*	212.9	202.61	*30.09*	34.74	32.257	22.65	*23.74*	22.228	68.78	69.77	*70.862*
Mk 08	20×10	585.142	572.823	*567.18*	*6.64*	17.94	11.957	4.624	*5.158*	4.086	*9.18*	7.7846	8.1701
Mk 09	20×10	426.732	378.553	*334.59*	54.01	*31.918*	40.435	*34.36*	31.107	32.518	*94.66*	89.18	92.919
Mk 10	20×15	*941.95*	1016.71	952.96	512.16	*304.73*	375.55	*302.92*	300.25	294.17	*432.55*	409.43	428.47
1a	10×5	2569.941	*2566.48*	2643.7	*36.1*	39.45	37.904	*28.513*	23.787	26.499	65.397	*68.216*	64.261
2a	10×5	2427.752	*2358.34*	2409.1	*263.23*	349.18	299.67	*128.801*	125.165	127.31	*495.1*	430.6	482.19
3a	10×5	*2540.038*	2781.75	2684.3	*344.798*	381.75	367.66	*237.84*	235.678	234.36	*840.107*	824.4	837.48
4a	10×5	2550.6519	*2496.806*	2529.3	107.73	108.03	*104.43*	62.82	73.2	*84.118*	*257.068*	234.362	223.79
5a	10×5	*2368.582*	2478.792	2511.2	*174.801*	280.5	221.93	126.849	*165.92*	147.96	386.09	*557.396*	421.78
6a	10×5	*2312.461*	2338.38	2326.8	*140.594*	154.11	152.47	*69.876*	61.56	64.685	*266.43*	242.667	207.66
7a	15×8	*2446.998*	2483.246	2473.6	*100.6*	124.73	119.58	59.655	*73.91*	71.882	*173.53*	149.5	163.98
8a	15×8	*2319.009*	2416.21	2392.6	*100.454*	113.51	102.01	*48.775*	38.146	42.894	*179.895*	165.25	175.83
9a	15×8	2206.967	2215.49	*2201.5*	*117.54*	127.974	115.8	*113.0686*	104.72	98.86	*378.748*	331.66	348.2
10a	15×8	*2458.58*	2478.9424	2469.6	93.285	*81.55*	84.325	51.95	53.768	*56.31*	176.418	162.435	*176.9*
11a	15×8	*2182.0226*	2257.72	2281.2	141.42	*112.32*	127.89	*100.3*	68.0005	84.869	*346.93*	212.62	201.74
12a	15×8	*2164.65*	2458.46	2208.4	*264.19*	292.63	271.26	*107.11*	94.74	103.17	*256.41*	217.72	228.17
13a	20×10	*3323.468*	3375.571	3340.5	*219.22*	306	300.34	*71.80629*	67.09	62.303	362.095	*374.108*	316.08
14a	20×10	*3172.51*	3211.188	3217.1	376.417	391.62	*344.79*	*526.51*	504.348	510.63	*817.54*	809.316	815.45
15a	20×10	*3249.22*	3512.75	3349	*457.15*	485.72	459.1	*317.02*	305.44	314.48	*241.37*	218.71	229.93
16a	20×10	*3002.78*	3643.46	3152.4	*413.71*	457.18	434.47	277.49	*304.27*	289.66	224.33	204.24	*235.48*
17a	20×10	*3011.3*	3126.245	3122.1	*247.29*	257.94	251.41	*269.2*	244.78	251.38	*514.75*	498.38	503.41
18a	20×10	*3261.75*	3456.68	3427.2	527.17	612.35	*518.76*	*129.81*	108.3	102.68	*421.74*	413.56	410.74

The best obtained values are marked in italic number

that MOPSO outweighs the NSGA-II algorithms in all metrics in terms of the number of optimum results out of 28 test problems. It is observed from that Table 4 that the proposed MOPSO superior to the NSGA-II and MOEA in 19, 18, 21, and 21 instances out of 28 test problems with respect to MID, RAS, SNS, and DM performance measures, respectively, for the objectives of makespan and mean flow time. Table 5 indicates that MOPSO performs superior to NSGA-II and MOEA in 21, 19, 21, and 17 instances out of 28 test problems in MID, RAS, SNS, and DM performance measures respectively, for the objective of makespan and mean tardiness.

In the present investigation, application of MOPSO results in large number of non-dominated solutions for optimization of objectives. The Pareto-optimal solutions obtained through MOPSO have been ranked by the composite scores obtained through maximum deviation theory (MDT) to choose the best solution. The decision matrix is normalized using the Eqs. 13 and 14. The objective weights are determined for the normalized values of objectives by applying maximum deviation method using Eqs. 15–22. The weighted objective values are estimated by multiplying the normalized objective values and the objective weights. The best solution is selected depending upon the composite scores obtained by addition of the all the weighted objective function values for each alternative. The objectives with highest composite score are chosen as the best solution. The solution ranking of the optimal solution set of problem 5a for makespan and mean flow time has been given in Table 6.

8 Conclusions

In this paper, benchmark instances from literature for flexible job shop scheduling problem are solved by an efficient multi-objective particle swarm optimization to find near-optimal schedules. The mutation operator generally used in genetic algorithm is embedded in MOPSO to avoid premature convergence and improve solution diversity. Further, maximum

Table 6 Solution ranking obtained through maximum deviation theory for the problem 5a

Run order	Objective function values		Normalized objective function values		Weighted objective function values		Composite score	Solution ranking
	Makespan (C_{max})	Mean flow time (f)	Makespan (N_{Cmax})	Mean flow time (N_F)	Makespan (WN_{Cmax})	WMean flow time (N_F)		
1	2300	1983.3	1	0	0.524	0	0.524	23
2	2324	1941.3	0.9155	0.1573	0.4797	0.0749	0.5546	20
3	2337	1880.1	0.8697	0.3865	0.4558	0.184	0.6397	14
4	2360	1865.4	0.7887	0.4416	0.4133	0.2102	0.6235	16
5	2364	1850.2	0.7746	0.4985	0.4059	0.2373	0.6432	11
6	2371	1843.7	0.75	0.5228	0.393	0.2489	0.6419	12
7	2373	1839.1	0.743	0.5401	0.3893	0.2571	0.6464	8
8	2380	1834.5	0.7183	0.5573	0.3764	0.2653	0.6417	13
9	2392	1812.9	0.6761	0.6382	0.3543	0.3038	0.658	4
10	2413	1797.9	0.6021	0.6944	0.3155	0.3305	0.646	9
11	2415	1788.5	0.5951	0.7296	0.3118	0.3473	0.6591	3
12	2418	1787.3	0.5845	0.7341	0.3063	0.3494	0.6557	5
13	2419	1782.1	0.581	0.7536	0.3045	0.3587	0.6631	2
14	*2421*	*1779.9*	*0.5739*	*0.7618*	*0.3008*	*0.3626*	*0.6634*	*1*
15	2443	1765.9	0.4965	0.8142	0.2602	0.3876	0.6477	7
16	2451	1756.4	0.4683	0.8498	0.2454	0.4045	0.6499	6
17	2460	1749.7	0.4366	0.8749	0.2288	0.4164	0.6452	10
18	2472	1741.7	0.3944	0.9049	0.2067	0.4307	0.6374	15
19	2519	1729.5	0.2289	0.9506	0.1199	0.4524	0.5724	17
20	2527	1728.2	0.2007	0.9554	0.1052	0.4548	0.5599	18
21	2530	1727	0.1901	0.9599	0.0996	0.4569	0.5565	19
22	2535	1724.2	0.1725	0.9704	0.0904	0.4619	0.5523	21
23	2548	1722.4	0.1268	0.9772	0.0664	0.4651	0.5315	22
24	2555	1721.6	0.1021	0.9801	0.0535	0.4665	0.52	24
25	2584	1716.3	0	1	0	0.476	0.476	25

The best obtained result is marked in italic number

deviation theory (MDT) is used to determine the weights of the attributes to develop a composite score to ease the decision-maker for selecting the best solution from a large set of Pareto solutions. The composite score for all the non-dominated solutions is obtained through summing the weighted objective values. The best solution is selected from all the non-dominated solution considering the highest composite score to avoid subjectiveness and impreciseness in the decision-making for the managers. This work offers an effective guideline to select optimum schedule for achieving the desired different objective simultaneously. From the comparative analysis, it can be concluded that the MOPSO algorithm is superior to NSGA-II and MOEA for different performance measures.

References

1. Garey EL, Johnson DS, Sethi R (1976) The Complexity of flow-shop and job-shop scheduling. Math Oper Res 1(2):117–129
2. Xia WJ, Wu ZM (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 48(2):409–425
3. Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput Ind Eng 54(3):453–473
4. Li JQ, Pan QK, Liang YC (2010) An effective hybrid tabu search algorithm for multi-objective flexible job shop scheduling problems. Comput Ind Eng 59(4):647–662
5. Kacem I, Hammadi S, Borne P (2002) Pareto-optimality approach for flexible job-shop scheduling problems, hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60(3–5): 245–276
6. Frutos M, Olivera AC, Tohme F (2010) A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. Ann Oper Res 181(1):745–765
7. Wang X, Gao L, Zhang C, Shao X (2010) A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. Int J Adv Manuf Technol 51(5–8):757–767
8. Moslehi G, Mahnam MA (2011) Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. Int J Prod Econ 129(1):14–22
9. Li JQ, Pan QK, Tasgetiren MF (2014) A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. Appl Math Model 38(3): 1111–1132
10. Rabiee M, Zandieh M, Ramezani P (2012) Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. Int J Prod Res 50(24):7327–7342
11. Li JQ, Pan QK, Chen J (2012) A hybrid Pareto-based local search algorithm for multi-objective flexible job shop scheduling problems. Int J Prod Res 50(4):1063–1078
12. Singh MR, Mahapatra SS (2011) A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks. Int J Adv Manuf Technol 62(1–4):267–277
13. Wang YM (1998) Using the method of maximizing deviations to make decision for multi-indices. Syst Eng Electron 20(7):24–26
14. Brandimarte P (1993) Routing and scheduling in a flexible jobshop by tabu search. Ann Oper Res 41(3):157–183
15. Perez MAF, Raupp FM (2014) A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem. J Intell Manuf. doi:10.1007/s10845-014-0872-0
16. Xing LN, Chen YW, Wang P, Zhao QS, Xiong J (2010) A knowledge-based ant colony optimization for flexible job shop scheduling problems. Appl Soft Comput 10(3):888–896
17. Bagheri A, Zandieh M, Mahdavi I, Yazdani M (2010) An artificial immune algorithm for the flexible job-shop scheduling problem. Futur Gener Comput Syst 26(4):533–541
18. Chang PC, Hsieh JC, Lin SG (2002) The development of gradual-priority weighting approach for the multi-objective flow-shop scheduling problem. Int J Prod Econ 79(3):171–183
19. Suresh R, Mohanasundaram KM (2006) Pareto archived simulated annealing for job shop scheduling with multiple objectives. Int J Adv Manuf Technol 29(1–2):184–196
20. Coello CA, Plido GT, Lechga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8(3):256–278
21. Lei D (2008) A Pareto archive particle swarm optimization for multi-objective job shop scheduling. Comput Ind Eng 54(4):960–971
22. Lei D, Wu Z (2006) Crowding measure-based multiobjective evolutionary algorithm for job shop scheduling. Int J Adv Manuf Technol 30(1–2):112–117
23. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proceedings of IEEE International Conference on Neural Network, Washington, USA, November/December 4:1942–1948
24. Yoshida H, Kawata K, Fukuyama Y, Nakanishi YA (2001) Particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans Power Syst 15(4): 1232–1239
25. Brandstatter B, Baumgartner U (2002) Particle swarm optimization mass-spring system analogon. IEEE Trans Magn 38(2):997–1000
26. Belmecheri F, Prins C, Yalaoui F, Amodeo L (2013) Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. J Intell Manuf 24(4):775–789
27. Basu M (2008) Dynamic economic emission dispatch using nondominated sorting genetic algorithm-II. Int J Electr Power Energy Syst 30(2):140–149
28. Kim B, Son S (2012) A probability matrix based particle swarm optimization for the capacitated vehicle routing problem. J Intell Manuf 23(4):1119–1126
29. Mostaghim S, Teich J (2003) Strategies for finding good local guides in multi-objective particle swarm optimization. Proceedings of the IEEE Symposium on Swarm Intelligence. 26–33
30. Wang LF, Singh C (2007) Environmental/economic power dispatch using a fuzzified multiobjective particle swarm optimization algorithm. Electr Power Syst Res 77(12):1654–1664
31. Li X (2004) Better spread and convergence: particle swarm multiobjective optimization using the maximin fitness. Lect Notes Comput Sci Genet Evol Comput - GECCO 3102:117–128
32. Raquel CR, Naval PC (2005) An effective use of crowding distance in multiobjective particle swarm optimization. Proceedings of Conference on Genetic and Evolutionary Computation (GECCO '05), New York, USA. 257–264
33. Alvarez-Benitez JE, Everson RM, Fieldsend JE (2005) A MOPSO algorithm based exclusively on pareto dominance concepts. Evol Multi-Criterion Optim Lect Notes Comput Sci 3410:459–473
34. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
35. Ebrahimipour V, Haeri A, Sheikhalishahi M, Asadzadeh SM (2012) Application of multi-objective particle swarm optimization to solve

a fuzzy multi-objective reliability redundancy allocation problem. J Saf Eng 1(2):26–38

36. Pant M, Radha T, Singh VP (2007) A simple diversity guided particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation (CEC), 25–28 September, Singapore, pp. 3294–3299
37. Dauzere-Peres S, Paulli EJ (1997) An integrated approach for modeling and solving the general multi-processor job shop scheduling problem using tabu search. Ann Oper Res 70:281–306
38. Bachlaus M, Pandey MK, Mahajan C, Shankar R, Tiwari MK (2008) Designing an integrated multi-echelon agile supply chain network: a hybrid taguchi-particle swarm optimization approach. J Intell Manuf 19(6):747–761
39. Modares H, Alfi A, Sistani MBN (2011) Parameter estimation of bilinear systems based on an adaptive particle swarm optimization. Eng Appl Artif Intell 23(7):1105–1111
40. Rahmati SHA, Zandieh M, Yazdani M (2013) Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. Int J Adv Manuf Technol 64(5–8):915–932
41. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: Methods and applications. Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH). Zurich, Switzerland