Sādhanā
DOI 10.1007/s12046-017-0611-2

A hybrid multi-objective evolutionary algorithm approach for handling sequence- and machine-dependent set-up times in unrelated parallel machine scheduling problem

V K MANUPATI[1,*], G RAJYALAKSHMI[1], FELIX T S CHAN[2] and J J THAKKAR[3]

[1]Department of Manufacturing, School of Mechanical Engineering, VIT University, Vellore, India
[2]Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[3]Department of Industrial and Systems Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India
e-mail: manupativijay@gmail.com; rajyalakshmimed@gmail.com; f.chan@polyu.edu.hk

MS received 26 December 2015; revised 10 July 2016; accepted 26 September 2016

Abstract. This paper addresses a fuzzy mixed-integer non-linear programming (FMINLP) model by considering machine-dependent and job-sequence-dependent set-up times that minimize the total completion time, the number of tardy jobs, the total flow time and the machine load variation in the context of unrelated parallel machine scheduling (UPMS) problem. The above-mentioned multi-objectives were considered based on non-zero ready times, machine- and sequence-dependent set-up times and secondary resource constraints for jobs. The proposed approach considers unrelated parallel machines with inherent uncertainty in processing times and due dates. Since the problem is shown to be NP-hard in nature, it is a challenging task to find the optimal/near-optimal solutions for conflicting objectives simultaneously in a reasonable time. Therefore, we introduced a new multi-objective-based evolutionary artificial immune non-dominated sorting genetic algorithm (AI-NSGA-II) to resolve the above-mentioned complex problem. The performance of the proposed multi-objective AI-NSGA-II algorithm has been compared to that of multi-objective particle swarm optimization (MOPSO) and conventional non-dominated sorting genetic algorithm (CNSGA-II), and it is found that the proposed multi-objective-based hybrid meta-heuristic produces high-quality solutions. Finally, the results obtained from benchmark instances and randomly generated instances as test problems evince the robust performance of the proposed multi-objective algorithm.

Keywords. Unrelated parallel machine; scheduling; meta-heuristics; NP-hard; MINLP.

1. Introduction

In the current competitive business environment, manufacturers face more challenges than ever before due to the highly volatile market. To remain competitive, efficient scheduling is one of the most critical issues in manufacturing and services, as described by Ying and Liao [1]. However, scheduling problems were first considered in the mid-1950s; since then, a variety of problems that deal with scheduling have been addressed and published in the literature. A comprehensive survey of the mentioned problems was reported by Allahverdi *et al* [2]. Parallel machine scheduling is a very common production environment that can be applied in many manufacturing and services systems and it has been extensively studied in the past few decades as discussed in Frendewey and Sumichrast [3].

A classical parallel machine scheduling problem (PMSP) was defined as a set of independent jobs to be processed on a number of available identical parallel machines, where each machine can process only one job at a time [4–7]. PMSPs are classified into three types: identical, uniform and unrelated by Cheng and Sin [8]. The identical machines can be defined as machines that are to be used for a task that has the same processing time across all the machines. In contrast, in uniform parallel machines, the processing time to perform a task may vary from one machine to another machine but each machine works at constant rate, as discussed by Santos *et al* [9]. Among the mentioned PMSPs, the unrelated parallel machine scheduling problem (UPMSP) represents a generalization of the other two categories. Here, a set of jobs can be processed on exactly one machine at different rates out of the set of parallel machines [5, 6].

*For correspondence
Published online: 21 February 2017

The evolution of UPMSP and the progress made with scheduling research were reported in Kamath [10] and Wang and Cheng [11]. Several researchers have addressed the UPMSP to improve various performance measures of the system. Li *et al* [12] examined the UPMSP to minimize the mean flow time, while Cao *et al* [13] studied the same problem by considering the total cost functions. Gairing *et al* [14] proposed a two-approximation algorithm to minimize the makespan for a set of independent jobs on an unrelated parallel scheduling problem. Handling of UPMSP with sequence-dependent times and machine set-up times is very difficult with exact methods due to the presence of several constraints like no job preemption, uninterrupted machining and non-linearity in nature, where the number of solutions obtained may not be feasible in polynomial amount of time. It is well known that such a problem is NP-hard in nature. Therefore it requires the application of evolution computing search techniques, with dominance and un-dominance methods, to find the Pareto optimal fronts for obtaining better solutions or near-optimal solutions. Rocha *et al* [15] developed a branch and bound algorithm to minimize multi-objectives such as makespan and total weighted tardiness as a linear combination for a sequence-dependent UPMSP. Fanjul-Peyro and Ruiz [16] proposed an iterative greedy local-search-based meta-heuristic to minimize the makespan of a set of independent jobs on unrelated parallel machines. Mehravaran and Logendran [17] addressed a UPMSP to minimize the bi-objective functions such as minimization of work-in-process inventory and maximization of customer service level in a supply chain. Lin *et al* [5, 6] proposed a genetic algorithm (GA) for UPMSPs by considering three objective functions including makespan, total weighted completion time and total weighted tardiness, separately, and compared it to the performance of various heuristics for proving its effectiveness. Yilmaz Eroglu *et al* [18] considered a UPMSP with sequence-dependent set-up times where evolutionary-algorithm-based GA with local search was used for minimizing the makespan.

Some studies have addressed different practical versions of UPMSPs by considering other features of real life scheduling problems, such as secondary constraints, non-zero ready times and so on, to bridge the gap between theoretical progress and industrial needs. This paper focuses on a UPMSP with machine-dependent and job-sequence-dependent set-up times, which has been addressed much less than the other PMSPs as noted by Yin *et al* [19]. The addressed problem has numerous industrial applications such as semiconductor manufacturing systems [20], automated gear manufacturing process [21], printed circuit board fabrication, textile manufacturing [22] and hospital operating rooms [23]. Chen and Wu [24] developed a heuristic method to minimize makespan in a UPMSP with different die types as a secondary resource constraint. Chen [25] developed an iterated local search to minimize the total weighted number of late jobs on UPMSP without preemption, with sequence-dependent set-up times and ready times. Ying *et al* [26] addressed the problem of minimizing the total tardiness by including the machine-dependent and sequence-dependent set-up times. Chen and Wu [24] solved a UPMSP with auxiliary equipment constraints such as secondary resource constraints. Hu *et al* [27] proposed a block-based erection scheduling problem in a shipyard, which is a case of parallel machine scheduling with precedence constraints and machine eligibility restrictions to minimize the makespan. Mehravaran and Logendran [17] proposed a sequence-dependent flow shop scheduling problem to minimize the performance measures of linear combined total weighted completion time and total weighted tardiness. Lamothe *et al* [28] proposed a new model to minimize the total tardiness by considering specific constraints such as secondary resources. Rambod and Rezaeian [29] addressed a UPMSP with rework processes, machine eligibility restrictions and sequence-dependent set-up times to minimize makespan. They proposed a new mixed integer non-linear programming model for small size problems and some meta-heuristics including GA and bees algorithm for medium and large size problems. Although, many researchers have investigated different problems in unrelated parallel machine scheduling, to the authors' knowledge, so far, considering sequence- and machine-dependent set-up times and secondary resource constraints for jobs simultaneously has not been properly addressed yet.

In this paper, a UPMSP with machine-dependent, sequence-dependent set-up times and secondary resource constraints (e.g., tools, labour, etc.) available at the same time is addressed. To improve the objective functions the performance measures such as minimization of makespan, tardiness, flow time and machine-load variation have been considered. To achieve this aim, the presented problem can be formulated as a fuzzy mixed-integer non-linear programming (FMINLP) model, and it belongs to the class of NP-hard problems. Since, the complexity of the problem is very high and multi-objective in nature, the need for effective multi-objective-based optimization techniques is employed. To solve the present problem CNSGA-II, MOPSO algorithms are adopted, and a novel hybrid AI-NSGA-II algorithm is proposed and compared. Moreover, a centroid-based distance method is used in the proposed algorithm to evaluate the fuzziness for better solutions. With several real-sized benchmark instances, the proposed model efficiency is proved.

In section 2, we provide a detailed description of the problem and its basic assumptions; later we also develop an FMINLP mathematical model along with the constraints. In section 3, we present a framework for the proposed evolutionary algorithms, i.e., AI-NSGA-II, NSGA-II and MPSO algorithm. Mapping of proposed algorithms on the research problem is explained in section 4. Experimentation with different small and large scale instances is illustrated, and the results are presented in section 5. The results

and discussion are detailed in section 6. Finally, conclusions are drawn, and future work delineated.

2. Problem description

In this paper, we introduced a UPMSP with non-zero ready times, job-sequence- and machine-dependent set-up times with auxiliary resource constraints in a fuzzy environment. The set $j = \{1, 2, \ldots, n\}$ of n jobs can be processed on set $M = \{1, 2, \ldots, m\}$ of m parallel machines, where each job needs one single task to complete the product with a known processing time. While processing the jobs on machines, it is necessary to take care of all the machines availability and then initiate the scheduled time at zero. We have considered sequence-dependent and machine-dependent set-up times, i.e., switching off machines from production and their set-up times are not considered until the completion of the first job on the same machine. Most of the real world problems are uncertain in nature, where the processing times and due dates greatly fluctuate. Therefore, it is necessary to introduce possibilistic data as triangular fuzzy numbers (TFNs) to estimate the parameters precisely; in section 4 a detailed description of the considered data and implementation are explained. We developed a novel multi-objective-based FMINLP model and considered objectives such as minimization of total makespan, total weighted flow time, total weighted tardiness and total machine load variation to examine the performance of the considered system. The above-mentioned problem assumptions are described as follows.

2.1 *Assumptions*

(a) Each job in a set requires an operation that can be processed on a set of all machines.
(b) Sequence- and machine-dependent set-up times for jobs are considered.
(c) Jobs may have different arrival (ready) times.
(d) Assignment of a job to a machine is permitted only if the required secondary resource(s) (e.g., tool, die) is (are) available.
(e) The machine processing should not be interrupted until a job operation is completed.
(f) Job pre-emption and machine breakdown are not allowed.
(g) Processing times and due dates of jobs are formulated in the form of TFNs due to a lack of knowledge of their precise parameters.

The notation and description used to formulate the above-mentioned problem mathematically are described in table 1. Decision variables are defined as follows:

Table 1. Notations and description.

Notation	Description
i	Job indices ($i = 1,2,3,\ldots, N$)
K	Machine index ($k = 1,2,3,\ldots,M$)
r	Required secondary resources index ($r = 1,\ldots,g$)
C_i	Run time of job i
s_i	Starting time of job i
d_i	Fuzzy due dates of job i
$C^*_{\max}$	Total completion time
$C^k_{\max}$	Largest completion time on machine k
s_r	Required secondary resource index ($s_r = 1,\ldots,S_R$)
P_{ijk}	Processing time of job i on machine k
S_{ijk}	Set-up time to switch from job i to job j on machine k
A_N	An arbitrary big number
T_i	Tardiness of job i
F_i	Flow time of job i

Decision variables

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

$$Y_{jk} = \begin{cases} 1 & \text{if job } i \text{ is assigned to machine } k \\ 0 & \text{otherwise} \end{cases}$$

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

2.2 *Mathematical model*

With the above-mentioned assumptions and notation, the problem is formulated as the following FMINLP model.

Objectives:

$$\text{Min } Z_1 = \sum_{i=1}^{N} C_i \tag{1}$$

$$\text{Min } Z_2 = \sum_{i=1}^{N} (C_i - S_i) \tag{2}$$

$$\text{Min } Z_3 = \sum_{i=1}^{N} (C_i - d_i) \tag{3}$$

$$\text{Min } Z_4 = \sum_{k=1}^{M} \left(C^*_{\max} - C^k_{\max}\right) \tag{4}$$

Subjected to constraints:

$$\sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{k=1}^{M} X_{ijk} = 1; \quad \forall j \tag{5}$$

$$\sum_{\substack{i=1 \\ i^1 j}}^{N} X_{ijk} = Y_{jk}; \quad \forall j,k \tag{6}$$

$$\sum_{\substack{j=1 \\ j\neq 1}}^{N} X_{ijk} \leq Y_{ik}; \quad \mathrm{i} = 1,\ldots,N \tag{7}$$

$$C_j^* + A_N\left(1 - X_{ijk}\right) \geq \mathrm{Max}\left\{C_i^* + S_{ijk,} S_j\right\} + P_{jk}; \quad \forall i,j,k \tag{8}$$

$$C_j^* + A_N\left(1 - Z_{ij}\right) \geq \mathrm{Max}\left(C_i^*, S_j\right) + \sum_{k=1}^{M} P_{jk} Y_{jk}; \quad \forall i, j \in S_r (i \neq j) \tag{9}$$

$$Z_{ij} + Z_{ji} = 1; \quad \forall i,j \in S_r, \quad i \neq j \tag{10}$$

$$\sum_{k=1}^{M} X_{ijk} \leq Z_{ij}; \quad \forall i, j \in S_r, \quad i \neq j \tag{11}$$

$$T_i \geq (C_i - d_i); \quad \forall i \tag{12}$$

$$C_i \geq 0; \quad T_i \geq 0; \quad \forall i, \quad X_{ijk}, \quad Y_{ik}, \quad Z_{ij} \in \{0,1\}; \quad \forall i,j,k. \tag{13}$$

The above-mentioned objectives, i.e., minimization of makespan (Z_1), flow time (Z_2), tardiness (Z_3) and machine load variation (Z_4), are given by Eqs. (1)–(4), respectively. Equation (5) assures that each job is assigned to only one position on a machine. Equation (6) suggests that if job j is allotted to machine k, then it is adopted by another job indicating dummy job 0. Constraint (7) specifies that at most one job can immediately adopt the previously allotted job i on machine k. Constraint (8) computes the completion time of a job when it is processed instantly after job i on the machine. Constraints (9 and 10) ensure that if job i and j need the same tool, one must be finished earlier before starting the other. Constraint (11) gives the relation between X_{ijk} and Z_{ij}. Equation (12) computes tardiness of job j. Equation (13) points out the non-negativity and integrality constraints.

3. Solution procedure

We first apply a conventional non-dominated sorting genetic algorithm-II (CNSGA-II), which includes non-dominated sorting and crowding distance operators for selection of the individuals to find the Pareto-optimal fronts [30]. With this algorithm, there might be a chance of obtaining premature convergence of the solutions. Therefore, a better algorithm is required to avoid the above-mentioned difficulties in achieving optimal Pareto optimal fronts. Hence, we adopt a multi-objective particle swarm optimization algorithm (MOPSO) to compare the assumed benchmark CNSGA-II algorithm. After that, we propose a novel artificial immunity-based non-dominated sorting genetic algorithm (AI-NSGA-II) in which the operators like vaccination and immunity selection greatly help in convergence of the solutions for approximating the Pareto-optimal frontiers. The above three meta-heuristics are used to generate effective and efficient solutions for the mentioned complex instances in the context of UPMSP. A detailed description of each algorithm with its frameworks as follows.

3.1 *Conventional non-dominated sorting genetic algorithm (CNSGA-II)*

First, the concept of the non-dominated sorting genetic algorithm (NSGA) was suggested by Goldberg [31] and Srinivas and Deb [32], who implemented it first. While using the NSGA, researchers found some drawbacks such as the following ones: (1) while operating a non-dominated sorting operator the algorithm takes more time for computation, (2) the absence of an elitism mechanism and (3) the absence of a tuneable parameter. Thus, the adopted CNSGA-II addresses the drawbacks of NSGA and details of the schematic procedure for processing CNSGA-II are shown in figure 1.

3.1a *Generation of fuzzy environment*: In this work, adopted and proposed evolutionary algorithms are used to find the Pareto-optimal frontier in a fuzzy domain to handle uncertainty in all parameters (e.g., processing time and due dates). Thus, it seems realistic to consider inherent fuzziness in the critical parameters. In this way, an imprecise processing time with some tolerance values (e.g., a processing time such as 40 ± 5 time units) could be modelled as a fuzzy number representing the incompleteness and imprecision of the required information. Ranking of fuzzy numbers to find the optimum solution is one of the important tasks to be achieved. According to Chen and Hwang [33], the ranking methods are classified into four main classes. Out of these classes, we adopt the centroid-based distance method that has been found from the literature to be best, where the ranking of fuzzy numbers is achieved with Euclidian distance from their centre of points to their origin [34]. A comprehensive method is used to represent each fuzzy parameter, i.e., a TFN. Moreover, the possibility distributions of the fuzzy processing times and due dates have been derived based on available historical data as well as subjective data about the knowledge and experience of the decision maker(s). Therefore, in this research work, we have considered processing times and due dates as fuzzy numbers. The objective functions (1–4) are reformulated based on

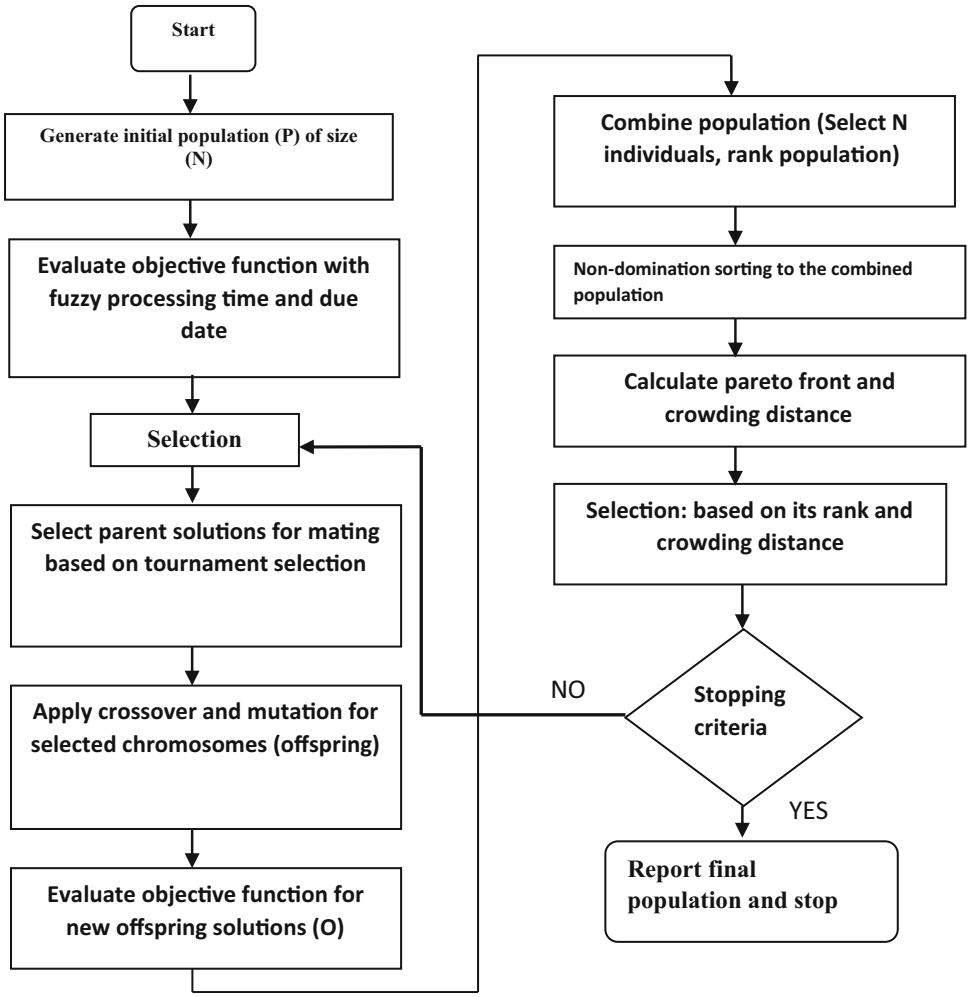

Figure 1. Schematic procedure for processing CNSGA II.

the concept of a triangular membership function to represent uncertainty.

Let us consider a triangular function with a lower limit a, an upper limit b and a value c, where $a < b < c$, as presented in figure 2. Membership functions represent a graphical representation of a fuzzy set by taking the universe of discourse on the x-axis and the degree of membership on the y-axis, as represented in Eq. (14) in the [0,1] interval.

$$\mu(x)=\begin{cases}0 & x\le a\\ (x-a)/(b-a) & a<x\le b\\ (b-a)/(b-c) & b<x<c\\ 0 & x\ge b\end{cases} \tag{14}$$

To face fuzzy constraints related to UPMS, the fuzziness of data is represented as in Torabi *et al* [35]. However, for our concerned problem, evolutionary algorithms are initialized in a state close to optimal solutions. Let us consider that $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$ are two TFNs;

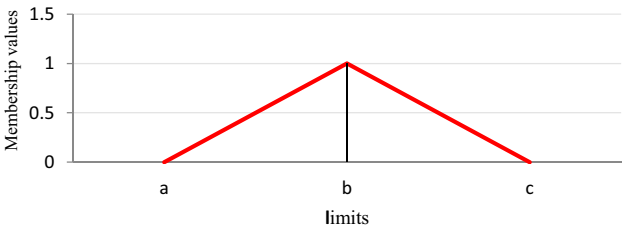

Figure 2. Representation of a triangular fuzzy number.

then the summation operator of the two fuzzy sets works according to the following formula:

$$\begin{aligned}A+B &= (a_1,a_2,a_3)+(b_1,b_2,b_3)\\ &= (a_1+b_1,a_2+b_2,a_3+b_3)\end{aligned} \tag{15}$$

According to the Zadeh [36] extension principle, the technique used for computing two or more membership

values, considering minimization and maximization of the fitness function, is represented as follows:

$$U_{A\cup B}(x) = \min(U_A(x), U_B(x)) \tag{16}$$

$$U_{A\cup B}(x) = \max(U_A(x),\ U_B(x)) \tag{17}$$

Therefore, the above operations are considered for calculation of starting and ending time of each job. We provide the following parametric definitions.

Interpretation 1 A fuzzy number P in parametric form is a pair $(\underline{p}, p')$ of function $\underline{P}(x)$, $P'(x)$, $0 \le x \le 1$ that satisfies all the following requirements:

1. $\underline{P}(x)$ is bounded by a left increasing continuous function,
2. $P'(x)$ is bounded by a left decreasing continuous function,
3. $\underline{P}(x) \le P'(x)$, $0 \le x \le 1$.

For the membership function $p = (a, b, c)$, let us assume a TFN with one defuzzifier b, the left fuzziness $a > 0$ and the right fuzziness $c > 0$ as presented earlier. In Eq. (18), the parametric form of the afore-mentioned membership function can be presented as follows:

$$P(x) = a(x-1) + b, \quad P'(x) = c(1-x) + b \tag{18}$$

Interpretation 2 For arbitrary fuzzy numbers $P = (\underline{P}, P')$ and $Q = (\underline{Q}, Q')$ the distance between P and Q can be calculated using the following formula:

$$D(P,Q) = \left[\int_0^1 (\underline{P}(x) - \underline{Q}(x))^2 dx + \int_0^1 (P'(x) - Q'(x))^2 dx\right]^{1/} \tag{19}$$

3.2 *Multi-objective particle swarm optimization (MOPSO)*

Kennedy *et al* [37] proposed the PSO algorithm for optimization. To solve the multi-objective problem, the PSO strategy has to be modified to find the Pareto-optimal front. Thus, PSO is suitable for multi-objective optimization with a high speed of convergence, allowing each individual to benefit from the experience. The main characteristic of this algorithm is to evaluate the performance measures of every particle and compare them with neighbourhood values, where all the non-dominated solutions generated at each iteration are stored in an external archive. A flowchart of MOPSO is shown in figure 3.

The schematic procedure followed by the MOPSO to obtain solution set, i.e., the Pareto optimal set, is as follows:

Step 1: Swarm (population) initialized.
Step 2: Generate random velocities.
Step 3: A set of leaders is initialized with some non-dominated solutions from the swarm, which are stored in the external archive.
Step 4: Some sort of quality measures such as velocity, position and cognitive learning factor are considered for all the leaders for selecting leaders in the archive.

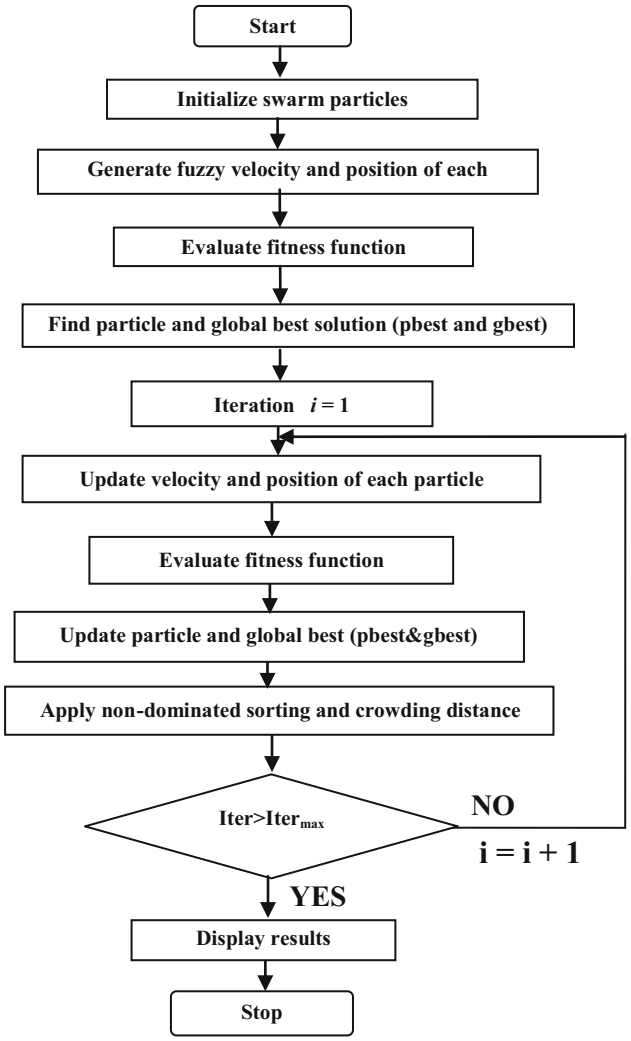

Figure 3. Flow chart of MOPSO.

Step 5: For each iteration update the velocity and position of each particle in the swarm.
Step 6: Evaluate the fitness function to sort *pbest* and *gbest*.
Step 7: Non-dominated sorting—set of optimal solutions where Pareto front is the best so that no front can dominate the existing one.
Step 8: Crowding distance—calculation of the distance between the members of each front to form the optimal Pareto front.
Step 9: Update leaders in the external archive, *gbest*.
Step 10: Check whether the termination (maximum iterations) criterion is reached. Otherwise, return to Step 2.

3.3 *Immunity-based hybrid non-dominated sorting genetic algorithm (AI-NSGA-II)*

We depict the main operations of the algorithm on the proposed problem in figure 4.

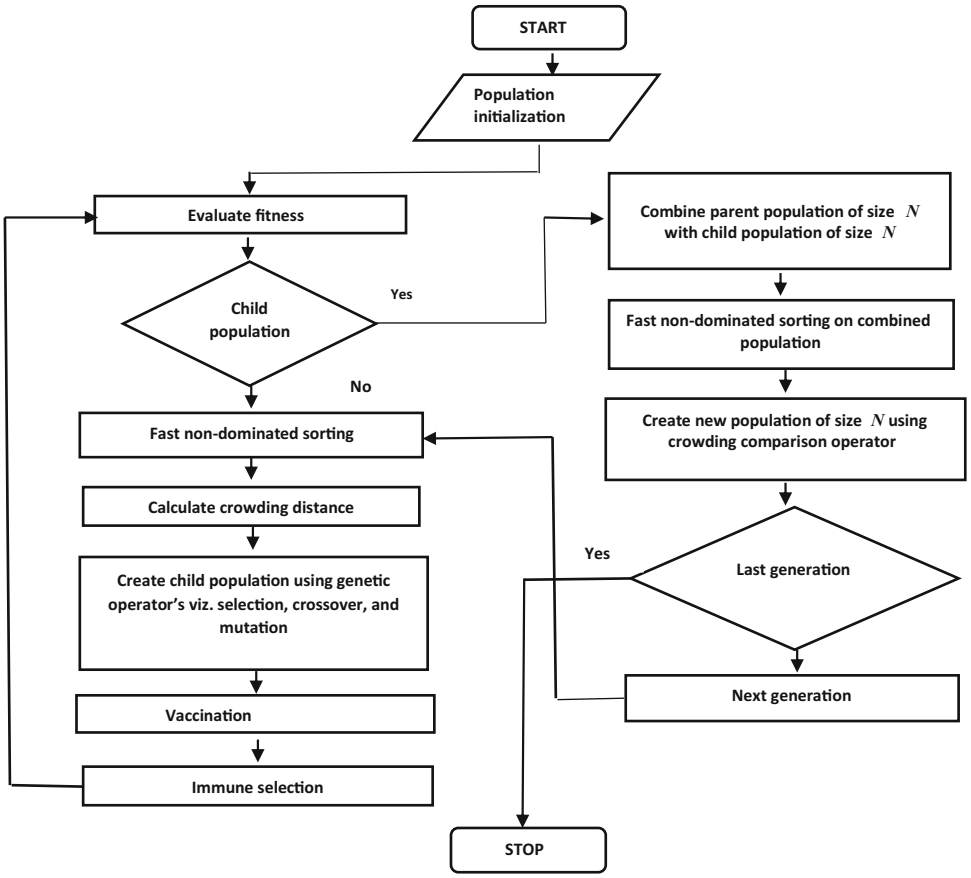

Figure 4. Flow chart of AI-NSGA-II.

Step 1. *Initialization*: Initialize the algorithm parameters, i.e., crossover probability (P_c), the mutation probability (P_m), the vaccination probability (P_v), the elimination ratio (R_{elim}), renewal rate ($a\%$), the memory set size (Memsize) and the termination conditions (Max_Gener, Improve_Gener).

In traditional GAs, each chromosome is generated randomly during the initialization of the population. With many experimental results, it is found that GA has a poor performance, and the result does not always show good solutions due to the random method. Therefore, most of the randomly generated chromosomes have a poor completion time and thus unfavourable traits in their offsprings. Reeves [38] presented a population initialization method that leads to the success of the GA. In this study, we used a similar type of initialization procedure for the proposed problem where processing of each job on assigning the machines may need a secondary resource simultaneously; it is presented in figure 5. While taking the operations into consideration, the job position value at each row is first sorted in ascending order and then the final sequence of jobs is obtained. If two jobs have common machines, then the previous job is given higher priority.

Step 2. *Parent selection*: Select pairs of parent chromosomes from the current population according to the selection probability and then apply the Rowlett wheel approach to generate the offsprings.

Step 3. The two-point crossover operator is used on the selected individuals with a crossover probability P_c to form new offsprings.

Step 4. *Mutation*: Perform mutation operation on the selected offspring based on the mutation probability P_m (we define the mutation probability not to each gene but to each offspring).

Step 5. *Vaccination*: Apply the vaccination principle on the selected offspring with the vaccination probability P_v. The vaccination, given an individual x, works under the principle of modifying the genes on some bits by prior knowledge so as to gain higher fitness with greater probability, as depicted in figure 6. This vaccination operation should satisfy the following conditions to increase the effectiveness of the algorithm. Firstly, the fitness evaluation of each gene bit of x is to find whether x is the optimal value; then the probability of transforming from x to x is 1. The vaccination on population $X = \{x1, x2,\ldots,xn\}$ is the operation carried out based on the vaccination probability in which individuals are selected

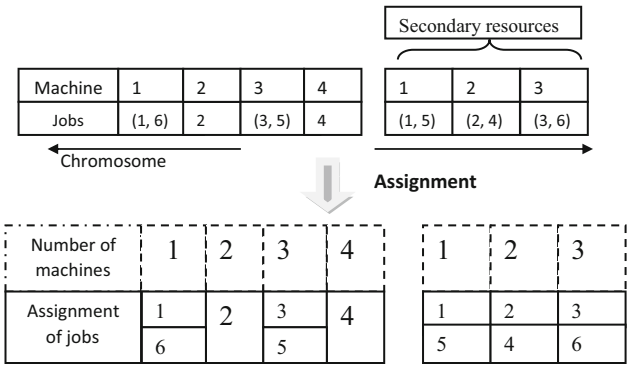

Figure 5. Illustration of chromosomes.

Let M_i be the best individual so far of the i^{th} population, σ a new similarity threshold between two individual and d a metric distance to measure affinity between two cells:

For *each MC_j of MS*

If *fitness (M_i) > fitness (MC_i) and $d(M_i, MC_i) > \sigma$*

Then if *MS (Memory Set) is not empty*

Then *M_i will be added to MS*

Else *substitute (M_i, MC_w) where MC_w is the worst memory cell in MS*

Else If *$d(M_i, MC_i) < \sigma$*

Then *substitute (M_i, MC_j)*

End For

Algorithm 1. The immuneselection principle.

from *X*. Vaccination is performed with prior knowledge of the performance of each step of the algorithm, i.e., operators add value in the functioning of the algorithm to a certain extent.

Step 6. *Immunity selection:* Update the memory set with the best individual of the current population. The following two steps accomplish it. Firstly, immunity testing, i.e., testing the antibodies as shown in algorithm 1. A population of individuals is optimized locally through fitness. A parent having smaller fitness values indicates that a series degeneration must have happened in the process of crossover or mutation. Moreover, the selection process is accomplished with following steps, i.e., parents with higher fitness will participate in the next competition to improve the performance; the second one is the annealing selection, i.e., each individual x_i in the present offspring $E_b = \{x_1, x_{2,\ldots,}x_n\}$ joins the new parents with a new probability $P(x_i)$.

Step 7. *Population renewal*: Update the population with the best fitness chromosome to avoid entrapment in local optimal and to obtain a global optimum.

Step 8. *Non-dominated sorting*: Set of optimal solutions where the Pareto front is the best, so no front can dominate the existing one.

Step 9. *Crowding distance*: Calculation of the distance (closeness) between the members of each front to form the optimal Pareto front.

Step 10. *Receptor editing*: Replace R_{elim} worst individuals in the current population with the highest makespan with new random individuals.

Step 11. *Termination test*: The algorithm terminates if a solution archives Max_gener generations or until the optimal solution is reached. If pre-specified iteration is satisfied, stop this algorithm. Otherwise, return to Step 2.

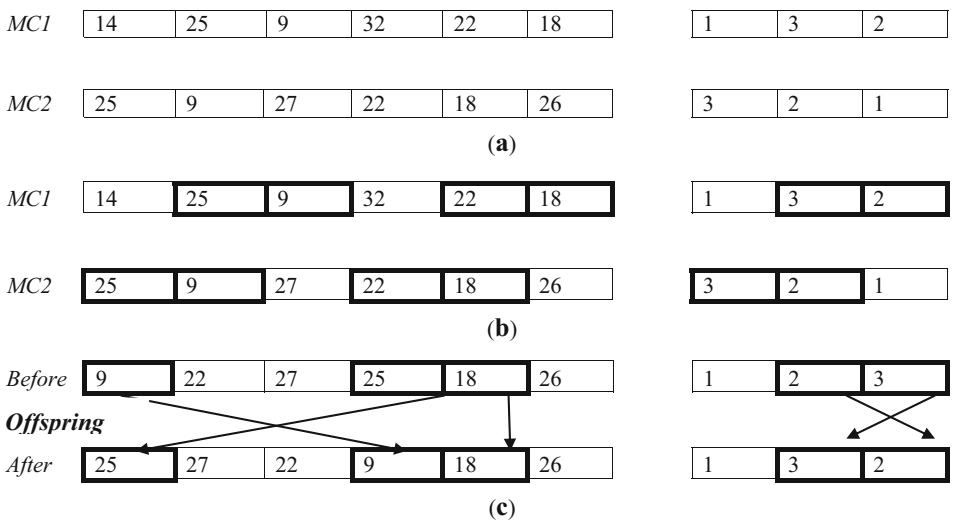

Figure 6. The vaccination principle. (**a**) The randomly selected memory cells. (**b**) The longest common sub-sequence is 25-9-22-18, 3-2. (**c**) The selected offspring before and after the vaccination.

4. Illustrative example

To solve the presented problem, some test problem instances, both small and large sizes of benchmark instances, and randomly generated instances following the uniform distribution for generating processing times, set-up times, job ready times and due date values, have been considered [4, 39].

Table 2 shows the detailed data of the considered 24 different problem instances with possibility triangular distributions, and it is denoted as $C' = (C^p, C^m, C^o)$, where C^p, C^m, and C^o are, respectively, the most pessimistic, the most possibilistic and the most optimistic values of C'. These values are applied for considering processing times and due dates. A possible value for processing times is generated using a continuous uniform distribution ranging from 5 to 50. For due dates, a possible value is generated using a continuous distribution $d_j = r_j + U(0,\beta)\min_{r=1}^{M}(\min_{i=1}^{N} AP_{ij}^{r})$ ranging in the interval 1–7. The β values in table 3 allow us to control the tightness and ready times $AP_{ij}^{k} = P_{jk} + S_{ij}; \quad (i,j = 1,2,\ldots,n)$ and $r = U(o,\alpha)$, while α makes it possible to control the arrival of the jobs.

Table 2. The performance measure of all methods with at different instances.

Problem instances	jJobs	mMachines	Card (g)	α	β
Problem information					
Scenario 1	6	4	2	1	3
Scenario 2	6	3	3	1	3
Scenario 3	6	2	2	1	3
Scenario 4	6	3	2	1	3
Scenario 5	8	3	3	1	3
Scenario 6	8	3	2	0.5	5
Scenario 7	8	4	3	0.5	5
Scenario 8	8	4	4	0.5	5
Scenario 9	8	5	3	0.5	5
Scenario 10	10	4	3	0.5	5
Scenario 11	10	3	3	0.5	5
Scenario 12	10	4	4	0.5	5
Scenario 13	25	8	4	0.5	3
Scenario 14	25	9	5	0.5	3
Scenario 15	25	10	6	0.5	3
Scenario 16	40	8	5	0.5	3
Scenario 17	40	9	5	0.5	3
Scenario 18	40	10	6	1	3
Scenario 19	50	12	6	1	3
Scenario 20	50	13	7	1	5
Scenario 21	70	12	6	1	5
Scenario 22	70	13	7	1	5
Scenario 23	90	15	6	1	5
Scenario 24	90	17	8	1	5

Table 3. The input parameters for NSGA-II, MOPSO and AI-NSGA-IIS.

Parameter	Value
NSGA II	
Population size	100 to –200
Total number of generations	150 to –300
Crossover probability	0.6 to –0.9
Mutation probability	0.01 to –0.1
MOPSO	
Cognitive factor ($c1$)	0.5 to –2
Social factor ($c2$)	0.5 to –2
Swarm size (N)	100 to –200
Number of iterations (K)	150 to –300
AI-NSGA II	
Population size	100 to −200
Total number of generations	150 to −300
Crossover probability	0.6 to −0.9
Mutation probability	0.01 to −0.1
Vaccination probability	0.8 to −0.9
Immuniszation probability	0.001 to −0.009

5. Computational results

Owing to the complexity of the problem, to generate the near-optimal solutions for multi-objectives more efficiently and effectively, AI-NSGA-II population-based random search algorithm is proposed to find the Pareto efficient solution. Hence, the input parameters of the considered three algorithms and their values are considered and they are presented in table 3. To evaluate the performance of the proposed algorithm, different benchmark instances are adopted. Figure 7 shows comparison results of different performance measures with the proposed algorithm and other adopted algorithms. For example, we have taken scenario 3 to detail the obtained results, where plots in figure 7a–c depict the performance measures of solutions with proper convergence and diversity. The X-axis indicates makespan as a performance measure and the Y-axis indicates flow time, tardiness and machine load variation as a performance measure, where the Pareto optimal curves for three different algorithms are shown with three different symbols. The dotted curve in figure 7 indicates Pareto optimal solutions for CNSGA-II; the triangular curve for MOPSO and the star symbol curve indicate AI-NSGA-II. It is evident from the Pareto optimal curves of shown figures that AI-NSGA-II shows better performance on both convergence and diversity compared with other algorithms.

Table 4 shows the results of performance measures such as makespan, flow time, tardiness and machine load variation with three different algorithms. In this work, the algorithms have been coded in MATLAB software, the problem is executed on a personal computer with Intel® Core™2 Duo CPU T7250 @2.00 GHz, 1.99 GB RAM, and the number of iterations is 300 for each algorithm to meet the termination criteria. The algorithm that can

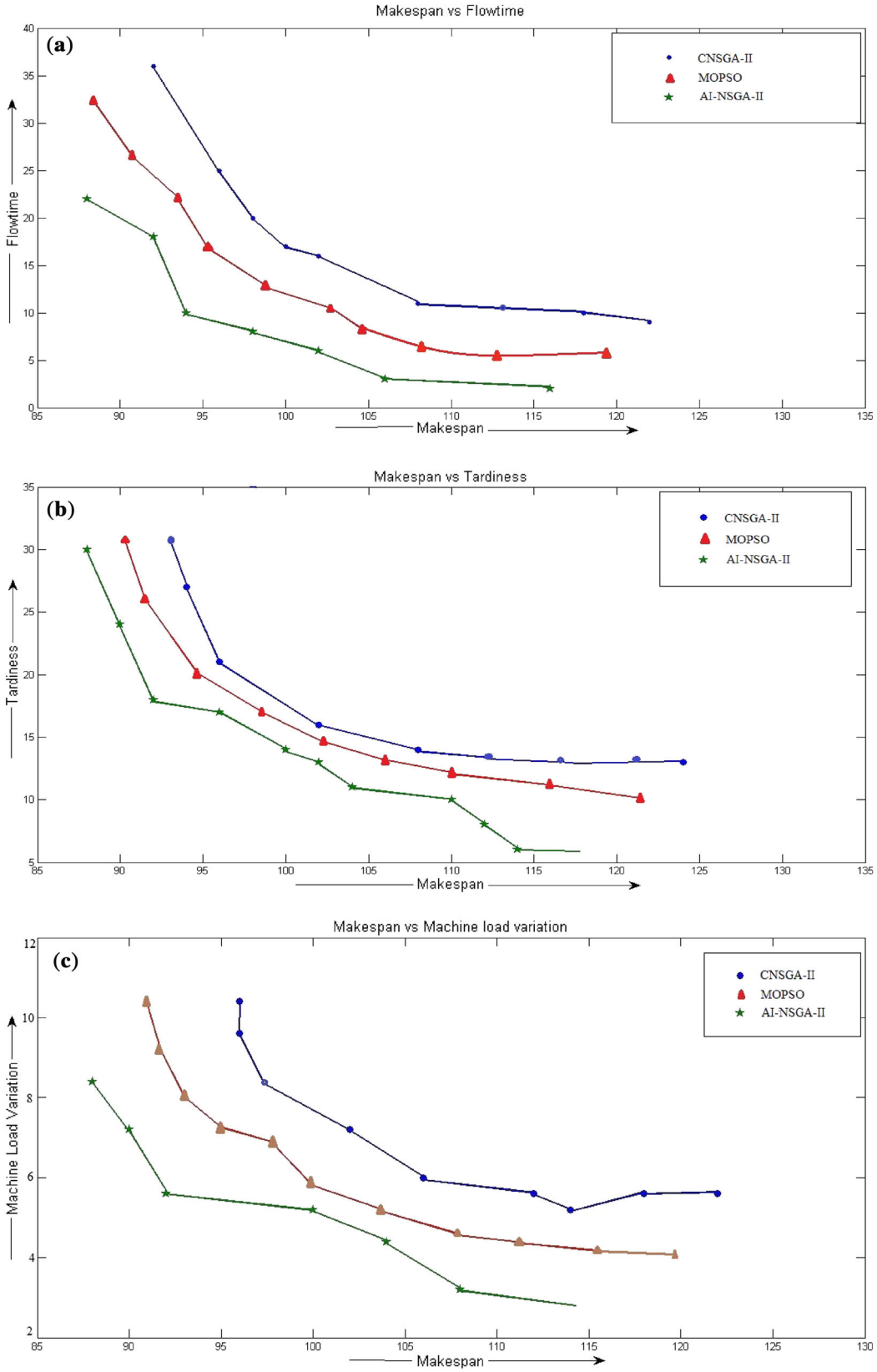

Figure 7. Performance measures. (**a**) (Mmakespan vs flowtime) results for proposed AI-NSGA-II, MOPSO and CNSGA-II algorithms for scenario 3. (**b**) (Mmakespan vs tardiness) results for proposed AI-NSGA-II, MOPSO and CNSGA-II algorithms for scenario 3. (**c**) (Mmakespan vs machine load variation) results for proposed AI-NSGA-II, MOPSO and CNSGA-II algorithms for scenario 3.

Table 4. The performance measures of all methods with different scenarios.

Problem instances	AI-NSGA-II				MOPSO				CNSGA-II			
	Makespan	Flow time	Tardiness	Machine load variation	Makespan	Flow time	Tardiness	Machine load variation	Makespan	Flow time	Tardiness	Machine load variation
Sce1	110	4	2	1	145	4	3	2	166	2	2	2
Sce2	98	5	3	3	109	3	1	3	142	3	3	1
Sce3	96	8	2	1	106	7	4	0	112	5	4	4
Sce 4	111	7	1	2	130	5	3	2	145	2	5	1
Sce 5	290	4	1	4	331	4	2	1	334	4	1	3
Sce 6	240	5	32	1	270	6	4	1	320	5	0	0
Sce 7	255	7	3	3	270	6	1	2	289	3	2	2
Sce 8	279	3	3	2	320	6	2	2	348	7	1	1
Sce 9	246	6	3	4	290	5	0	1	320	2	1	1
Sce 10	310	2	4	2	386	2	1	3	394	6	0	4
Sce 11	289	4	1	2	325	2	3	0	353	3	1	1
Sce 12	354	8	3	2	396	4	1	2	403	2	2	0
Sce 13	399	9	2	3	423	3	2	1	443	4	0	2
Sce 14	379	5	2	1	410	8	2	2	432	3	1	3
Sce 15	368	8	3	4	408	6	1	1	428	7	1	1
Sce 16	479	3	1	3	513	7	4	1	549	8	2	2
Sce 17	421	5	2	5	458	11	2	0	495	4	0	0
Sce 18	455	9	3	4	486	2	0	1	497	7	1	3
Sce 19	399	3	1	2	427	5	1	3	436	3	1	4
Sce 20	386	7	0	4	406	9	3	2	432	6	0	1
Sce 21	589	9	1	4	631	9	2	2	681	7	2	2
Sce 22	510	5	3	5	536	10	1	3	573	9	2	1
Sce 23	498	8	0	3	529	6	1	0	592	5	0	0
Sce 24	555	4	1	2	597	9	2	2	633	7	1	1

converge quickly compared with other algorithms can be considered as good. Finally, from the experimental results, we find that the proposed AI-NSGA-II performs better than MNSGA-II and MOPSO.

6. Conclusions and recommendations for future studies

This paper addressed a multi-objective-based unrelated PMSP with machine-dependent and job-sequence-dependent set-up times. With objectives such as makespan, flow time, tardiness and machine load variation the behaviour of the system is analysed to improve the systems performance. A fuzzy mixed-integer non-linear programming model is developed to solve the problem optimally. To deal with the above-mentioned multi-objective mixed-integer non-linear programming problem and to find the quality solutions, a multi-objective-based evolutionary algorithm (MOEA) approach is adopted and an effective AI-NSGA-II algorithm is proposed to find the Pareto-optimal frontier. Dominance and un-dominance method is critical for MOEA to find the Pareto optimal fronts; for obtaining better solutions and to cater to fuzzy environment a centroid-based distance method is used. Several benchmark instances are presented to validate the proposed method. The performance of the proposed algorithm AI-NSGA-II is compared to those of adopted conventional MNSGA-II and MPSO algorithms. Experimental results demonstrated the effectiveness of the proposed AI-NSGA-II in the comparison with the existing MNSGA-II and MPSO algorithms in overall performance indicators while solving benchmark instances. The future studies may include consideration of many objectives that involve different performance criteria to predict the real-time behaviour of a system since existing research efforts are directed at solving only single and multi-objective criterion problems.

References

[1] Ying K C and Liao C J 2004 An ant colony system for permutation flow-shop sequencing. *Comput. Oper. Res.* 31(5): 791–801

[2] Allahverdi A, Ng C T, Cheng T E and Kovalyov M Y 2008 A survey of scheduling problems with setup times or costs. *Eur. J. Oper. Res.* 187(3): 985–1032

[3] Frendewey J O and Sumichrast R T 1988 Scheduling parallel processors with setup cost and resource limitations. *Decision Sci.* 19(1): 138–146

[4] Blidgue U, Kirac F, Kurtulan M and Pekgun P 2004 A tabu search algorithm for parallel machine total tardiness problem. *Comput. Oper. Res.* 31(3): 397–414

[5] Lin S W, Lu C C and Ying K C 2011 Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *Int. J. Adv. Manuf. Technol.* 53(1–4): 353–361

[6] Lin Y K, Pfund M E and Fowler J W 2011 Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Comput. Oper. Res.* 38(6): 901–916

[7] Tavakkoli-Moghaddam R, Taheri F, Bazzazi M, Izadi M and Sassani F 2009 Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *Comput. Oper. Res.* 36(12): 3224–3230

[8] Cheng T C E and Sin C C S 1990 A state of the art review of parallel-machine scheduling research. *Eur. J. Oper. Res.* 47(3): 271–292

[9] Santos F Charrua, Francisco Brojo and Pedro M Vilarinho 2012 *Lot sizing and scheduling in parallel uniform machines—a case study*. INTECH Open Access Publisher. doi:10.5772/50975

[10] Kamath S 2011 Unrelated parallel machine scheduling—perspectives and progress. *OPSEARCH* 48(4): 318–334

[11] Wang X and Cheng T C E 2015 A heuristic for scheduling jobs on two identical parallel machines with a machine availability constraint. *Int. J. Prod. Econ.* 161: 74–82

[12] Li K, Shi Y, Yang S L and Cheng B Y 2011 Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. *Appl. Soft Comput.* 11(8): 5551–5557

[13] Cao D, Chen M and Wan G 2005 Parallel machine selection and job scheduling to minimize machine cost and job tardiness. *Comput. Oper. Res.* 32(8): 1995–2012

[14] Gairing M, Monien B and Woclaw A 2007 A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theor. Comput. Sci.* 380(1): 87–99

[15] Rocha P, Ravetti M, Mateus G and Pardalos P 2008 Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Comput. Oper. Res.* 35: 1250–1264

[16] Fanjul-Peyro L and Ruiz R 2010 Iterated greedy local search methods for unrelated parallel machine scheduling. *Eur. J. Oper. Res.* 207: 55–69

[17] Mehravaran Y and Logendran R B 2011 Supply chain scheduling on unrelated-parallel machines. *J. Chin. Inst. Ind. Eng.* 28(2): 91–101

[18] Yilmaz Eroglu D, Ozmutlu H C and Ozmutlu S 2014 Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times. *Int. J. Prod. Res.* 52(19): 5841–585

[19] Yin N, Kang L, Sun T C, Yue C and Wang X R 2014 Unrelated parallel machines scheduling with deteriorating jobs and resource dependent processing times. *Appl. Math. Model.* 38(19): 4747–4755

[20] Wang I L, Wang Y C and Chen C W 2013 Scheduling unrelated parallel machines in semiconductor manufacturing by problem reduction and local search heuristics. *Flex. Serv. Manuf. J.* 25(3): 343–366

[21] Gokhale R, and Mathirajan M 2012 Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing. *Int. J. Adv. Manuf. Technol.* 60(9–12): 1099–1110

[22] Chang P C and Chen S H 2011 Integrating dominance properties with genetic algorithms for parallel machine scheduling problems with setup times. *Appl. Soft Comput.* 11(1): 1263–1274

[23] Vairaktarakis G L and Cai X 2003 The value of processing flexibility in multipurpose machines. *IIE Trans.* 35(8): 763–774

[24] Chen J F and Wu T H 2006 Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega* 34(1): 81–89

[25] Chen J F 2005 Unrelated parallel machine scheduling with secondary resource constraints. *Int. J. Adv. Manuf. Technol.* 26(3):285–292

[26] Ying K C, Lee Z J, Lu C C and Lin S W 2012 Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups. *Int. J. Adv. Manuf. Technol.* 58(5–8): 671–682

[27] Hu X, Bao J S and Jin Y 2010 Minimising makespan on parallel machines with precedence constraints and machine eligibility restrictions. *Int. J. Prod. Res.* 48(6): 1639–1651.

[28] Lamothe J, Marmier F, Dupuy M, Gaborit P and Dupont L 2011 Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints. *Comput. Oper. Res.* 39(6): 1236–1244

[29] Rambod M and Rezaeian J 2014 Robust meta-heuristics implementation for unrelated parallel machines scheduling problem with rework processes and machine eligibility restrictions. *Comput. Ind. Eng.* 77: 15–28

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

[31] Goldberg D E 1989 *Genetic algorithms in search optimization and machine learning*. New York: Addison-Wesley

[32] Srinivas N and Deb K 1994 Multi-objective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* 2: 221–248

[33] Chen P C and Hwang Y K 1992 SANDROS: A motion planner with performance proportional to task difficulty. In: *Proceedings of the International Conference on Robotics and Automation, IEEE* 1992, pp. 2346–2353

[34] Cheng R and Gen M 1998 Loop layout design problem in flexible manufacturing systems using genetic algorithms. *Comput. Ind. Eng.* 34(1): 53–61

[35] Torabi S A, Sahebjamnia N, Mansouri S A and Bajestani M A 2013 A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Appl. Soft. Comput.* 13(12): 4750–4762

[36] Zadeh LA (1965) Fuzzy sets. *Inf. Control* 8(3): 338–353

[37] Kennedy J F, Kennedy J, Eberhart R C and Shi Y 2001 *Swarm intelligence*. Morgan Kaufmann, San Francisco

[38] Reeves C R 1995 A genetic algorithm for flowshop sequencing. *Comput. Oper. Res.* 22(1): 5–13

[39] Huo Y, Leung J Y T and Zhao H 2007 Bi-criteria scheduling problems: number of tardy jobs and maximum weighted tardiness. *Eur. J. Oper. Res.* 177(1): 116–134