Journal of Manufacturing Systems xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling

Nasser Shahsavari-Pour[a], Behrooz Ghasemishabankareh[b,*]

[a] Department of Industrial Management, Vali-e-Asr University, Rafsanjan, Iran
[b] Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Kerman, Iran

ARTICLE INFO

Article history:
Received 31 March 2012
Received in revised form
18 December 2012
Accepted 22 April 2013
Available online xxx

Keywords:
Flexible job shop scheduling problem
Pareto optimal solution
Genetic algorithm
Simulated annealing
Multi objective genetic algorithm

ABSTRACT

Finding feasible scheduling that optimize all objective functions for flexible job shop scheduling problem (FJSP) is considered by many researchers. In this paper, the novel hybrid genetic algorithm and simulated annealing (*NHGASA*) is introduced to solve FJSP. The *NHGASA* is a combination of genetic algorithm and simulated annealing to propose the algorithm that is more efficient than others. The three objective functions in this paper are: minimize the maximum completion time of all the operations (makespan), minimize the workload of the most loaded machine and minimize the total workload of all machines. Pareto optimal solution approach is used in *NHGASA* for solving FJSP. Contrary to the other methods that assign weights to all objective functions to reduce them to one objective function, in the *NHGASA* and during all steps, problems are solved by three objectives. Experimental results prove that the *NHGASA* that uses Pareto optimal solutions for solving multi-objective FJSP overcome previous methods for solving the same benchmarks in the shorter computational time and higher quality.

1. Introduction

Classical job shop scheduling (JSP) can be stated as follows: there are m different machines and n different jobs to be scheduled. Each job is composed from a set of operations, and the operation order on machine is pre-specified. Each operation is characterized by required machine and the fixed processing time. All machines are available in the starting time and also all jobs can be processed at time 0. JSP is well-known to be Np-hard [1]. Nowadays, in the highly competitive environment of industries, if all requirements want to be satisfied, flexible systems should be utilized with high efficiency. FJSP is a generalization of the JSP [2]. FJSP can be separated in two sub-problems [3]: 1. Routing sub-problem that assigns each operation to available machines. 2. Scheduling sub-problem that define the operation sequences of all the jobs. These two sub-problems are used to generate the feasible solution for FJSP to optimize the objective functions. Brucker and Schile [4] propose a polynomial algorithm for two-job problems. FJSP is extremely harder than JSP. FJSP is NP-hard and combinatorial as well. Therefore, it cannot be solved by exact methods [1] because the computation time increases exponentially. In the recent years, a lot of researches have been carried out to propose methods that can solve FJSP in the reasonable computation time and can find optimal or approximate optimal solution. In particular, due to complexity of the JSP and FJSP, meta-heuristic methods are generally proposed such as tabu search (TS) [5], simulated annealing (SA) [6], particle swarm optimization (PSO), ant colony optimization (ACO) [7], GA [8], and immune algorithm [9].

During solving FJSP in the real cases, hierarchical approach and integrated approach have been used [3]. The hierarchical approach makes the problem easier by dividing the FJSP into a sequence of sub-problems. Brandimarte [10] proposed the hierarchical approach for the FJSP. In hierarchical approach assigning operations to machines and also defining job sequences are considered as two separate problems. Meanwhile, assigning operations to machines and also defining job sequences are being considered simultaneously.

Different approaches of TS for solving FJSP were proposed in Hurink et al. [11] and Dauzère-Pérès's paper [12]. Due to, GA is one of the most effective methods for combinatorial optimization problems, therefore, most of FJSP were solved by GA. By altering GA or combining this method with other algorithms it can be more effective for solving FJSP. Each of these methods is varied in different aspects such as encoding scheme, dispatching rules, generating initial population, creating offspring, and objective functions. Yang [13] proposed the algorithm based on genetic algorithm and discrete dynamic programming. Also multistage genetic algorithm proposed in [14] to solve FJSP. Ref. [15] solved the multi-objective FJSP by using dispatching rules discovered through

* Corresponding author. Tel.: +98 937 2762671; fax: +98 341 2131780.
E-mail addresses: Shahsavari_n@alum.sharif.edu (N. Shahsavari-Pour), Behrooz_ghasemi_sh@yahoo.com (B. Ghasemishabankareh).

0278-6125/$ – see front matter © 2013 Published by Elsevier Ltd on behalf of The Society of Manufacturing Engineers.
http://dx.doi.org/10.1016/j.jmsy.2013.04.015

genetic programming. An effective bi-population based estimation of distribution algorithm (BEDA) was proposed to solve FJSP in [16]. Zheng et al. [17] proposed a model for FJSP with transportation constraints and bounded processing times. Ulungu et al. [18] used SA and Wu and Xia [19] combine PSO and SA to solve FJSP. An improved GA combined with TS in [20] and also PSO combined with TS as local search in [21] was proposed to solve the multi objective FJSP. In [22] a variable neighborhood search (VNS) algorithm based on integrated approach was proposed to solve bi-criteria FJSP.

In this paper, the novel hybrid meta-heuristic (*NHGASA*) algorithm is introduced to solve multi objective FJSP. The *NHGASA* algorithm decreases computation time and extremely increases quality of the solutions for multi objective FJSP. This novel hybrid meta-heuristic (*NHGASA*) algorithm is the combination of GA and SA. Whereas previous methods were forced to run the algorithm more than once to obtain a set of optimal solutions, *NHGASA* is using Pareto optimal solution in its process, therefore, in the last generation the set of optimal solutions is calculated. Obviously this aspect can save time. This is one of the most important aspects of the *NHGASA*. The three objectives are considered to minimize: makespan, critical machine workload, and total workload of all machines. While other methods assign weights to each objective function and change it to a single objective problem, another innovation of this paper is using efficiently Pareto optimal solution (multi-objective genetic algorithm (MOGA)) to solve FJSP with three objectives from the beginning to the end. The experimental results prove that the multi-objective results of *NHGASA* algorithm overcome other approaches to solving the same benchmarks in the shorter computation time and higher quality.

The paper is organized as follows. Section 2 gives assumption and the mathematical formulation of FJSP. Section 3 describes solving method of FJSP that shortly describes the MOGA concepts used for FJSP and proposing *NHGASA* method. Section 4 analyzes the performance results of *NHGASA* when applied to solve common benchmarks in literature. Finally, Section 5 gives some concluding remarks and directions for future work.

2. Mathematical formulation

The FJSP considers n jobs to be processed on m machines while minimizing the objective functions. There is a set of n jobs $J = \{1, 2, 3, \ldots, n\}$ and m machines $M = \{1, 2, 3, \ldots, m\}$. The assumptions in the FJSP are as follows:

(1) All machines are available at time 0.
(2) All jobs can be started at time 0.
(3) Each machine can process only one operation at a time.
(4) Once an operation begins on a machine, it must not be interrupted.
(5) The order of operations for each job is pre-specified.
(6) Neither release times nor due dates are specified.
(7) Job transportation time among machines is not considered.

Before describing the mathematical model it is better to define some symbols and notifications as follows:

K a set that defines number of operations for each job. If $K(i) = j$, it defines that job j has i operations
$C_{K(i)ij}$ completion time of $K(i)$th operation of job i on machine j
t_{lij} processing time of lth operation of job i on machine j
T_j the sum of the time that machine j is doing process

The three objective functions (which have been used to evaluate the efficiency of *NHGASA* algorithms in solving multi-objective FJSPs) can be described as the following. Eq. (1) gives the first objective makespan and also means to minimize the maximum finishing time considering all the operations. Eq. (2) gives the second objective which is to minimize the workload of the most loaded machine. Eq. (3) gives the third objectives that minimize the total workload of machines:

$$F1: \quad \text{Min}\max\{C_{k(i)ij}\}; \quad K(i) \in K; \quad i \in J; \quad j \in M \tag{1}$$

$$F2: \quad \text{Min}\max\left\{T_j = \sum_{i=1}^{n}\sum_{l=1}^{k(i)} t_{lij}\right\} \tag{2}$$

$$F3: \quad \text{Min}\left\{\sum_{j=1}^{m} T_j\right\} \tag{3}$$

s.t.

$$C_{lij} - t_{lij} \geq C_{(l-1)ip}; \quad L = 1, 2, \ldots, K(i); \quad i = 1, 2, \ldots, n; \quad p, j = 1, , 2, \ldots, m \tag{4}$$

$$C_{lij} - t_{lij} \geq C_{QEj}; \quad Q, L = 1, 2, \ldots, K(i); \quad E, i = 1, 2, \ldots, n; j = 1, 2, \ldots, m \tag{5}$$

$$C_{lij} \geq 0; \quad L = 1, 2, \ldots, K(i); \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m \tag{6}$$

All objective functions are minimization. The two basic processing constraints are shown in Eqs. (4) and (5). The precedence constraints among operations of the same job should be followed (Eq. (4)). Each machine becomes available to other operations when the operations currently assigned are completed (Eq. (5)).

3. Solution procedure

Before defining the solution procedure it is better to briefly describe the MOGA approach.

3.1. MOGA procedure

One of the most useful meta-heuristic algorithms for solving multi-objective optimization is GA. About, 90% of multi-objective optimization algorithms used Pareto front to solve problems and more than 70% meta-heuristics algorithms were related to evolutionary algorithms [23]. There are a lot of different approaches to solving multi objective optimization by GA. In this paper, MOGA [24] is used efficiently to solve multi objective FJSP.

MOGA [25] was the first multi-objective GA that clearly used ranking of solutions and niching approach at the same time to search efficiently in the true Pareto front while existing diversity in the population. Different steps of MOGA are given as follows:

Step 1. Ranking the population

This method of ranking was proposed by Fonseca and Fleming [25]. In this method, the ranking of each solution (x) in the (tth) generation is calculated by Eq. (7). $DS(x,t)$ is the number of solutions that dominate x in the tth generation.

$$R(x, t) = 1 + DS(x, t) \tag{7}$$

Step 2. Calculating fitness value

According to first step, the fitness value is evaluated for each solution (x) in the (tth) generation by Eq. (8).

$$fit(x,t) = N - \sum_{k=1}^{R(x,t)-1} n_k - [0.5 \times (n_{R(x,t)} - 1)] \quad (8)$$

N is the population size; $R(x,t)$ is the ranking of solution x in tth generation; n_k is the number of solution with rank k; $n_{R(x,t)}$ is the number of solution with rank $R(x,t)$.

Step 3. Calculating distance between two points in the solution space

Calculate distances between all pairs of solutions. The distance between solutions x and y is computed by Eq. (9).

$$Dis_F(x,y) = \sqrt{\sum_{k=1}^{K} \left(\frac{F_k(x) - F_k(y)}{F_k^{\max} - F_k^{\min}} \right)^2} \quad (9)$$

$F_k^{\max}$ is the maximum value of objective function F_k in each population; $F_k^{\min}$ is the minimum value of objective function F_k in each population; K is the number of objective functions.

Step 4. Calculating niche count

According to Eq. (9) the niche count for each solution (x) in the (tth) generation of each population (P) is computed by Eq. (10). σ_{share} is niche size. Ref. [26] proposed effective approach to estimate and update σ_{share}.

$$nc(x,t) = \sum_{y \in P, R(y,t)=R(x,t)} \max \left\{ \frac{\sigma_{share} - Dis_F(x,y)}{\sigma_{share}}, 0 \right\} \quad (10)$$

Step 5. Calculating modified fitness value

Modified fitness value for each solution (x) is calculated by Eq. (11).

$$fit'(x,t) = \frac{fit(x,t)}{nc(x,t)} \quad (11)$$

Step 6. Calculating the normalized fitness

Normalized fitness is calculated by Eq. (12).

$$fit''(x,t) = \frac{fit'(x,t) n_R(x,t)}{\sum_{y \in Pt, R(y,t)=R(x,t)} fit'(y,t)} fit(x,t) \quad (12)$$

At the end of 6th step normalized fitness is used in the selection procedure in GA.

3.2. Procedure of NHGASA

The *NHGASA* is a novel hybrid algorithm. This algorithm is the combination of GA and SA. Compared to the other algorithms the *NHGASA* has shorter computation time and higher quality of solutions. The frame work of *NHGASA* is given as follows; first initial population is generated by GA then exact number of individuals in each population is improved by SA. If there is no improvement in SA algorithm, in definite iterations, then a method of altering in each individual is changed (Fig. 3). This is the most important and sensitive aspect about SA in the *NHGASA* algorithm. The general framework of the *NHGASA* is shown in Fig. 1.

Now the procedure of *NHGASA* is given as follows:

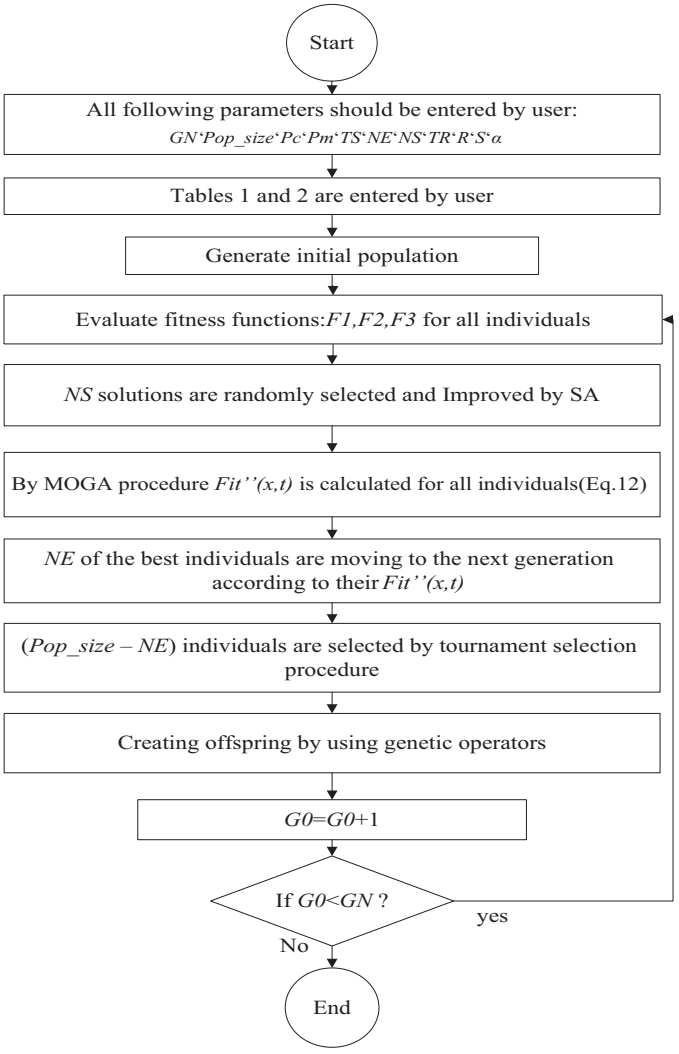

Fig. 1. Basic frame work of NHGASA.

Step 1. Representation

For representing approaches three critical issue emerged concerning with encoding and decoding between chromosomes and solutions. 1. Feasibility of chromosome, it means that whether a solution decoded from a chromosome lies in the feasible region of the given problem. 2. Legality of a chromosome, it means that whether a solution decoded from a chromosome does not lie outside of solution space. 3. Uniqueness of chromosome, it means after decoding chromosome to the solutions only one solution should be obtained from each chromosome and vice versa. After solving the FJSP, the sequences of the operations should be identified and the machines that process the operations should be defined as well. Therefore, each schedule needs two chromosomes. The representation method that is used in this paper is captured from [21]. In this method, each solution consists of two parts: parts A and B. Part A, represents the machines that process the operations and part B, illustrates a sequence of the operations. The number of genes in each part is equal to the sum of the operations in all jobs. For part A each gene value shows machine's number and in part B shows the job's number. It should be considered that number of each job that is repeated in part B must be equal to the number of that job's operations. Fig. 2 illustrates the sample chromosome for the example of 5-machines and 4-jobs that is presented in Table 1.

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015

Table 1
4 × 5 with 12 operations problem.

Job	Operation	Processing time for machine *Mi*				
		M1	*M2*	*M3*	*M4*	*M5*
J1	1	2	5	4	1	2
	2	5	4	5	7	5
	3	4	5	5	4	5
J2	1	2	5	4	7	8
	2	5	6	9	8	5
	3	4	5	4	54	5
J3	1	9	8	6	7	9
	2	6	1	2	5	4
	3	2	5	4	2	4
	4	4	5	2	1	5
J4	1	1	5	2	4	12
	2	5	1	2	1	2

As shown in Fig. 2 parts A is consisted of machine's number $M = \{1, 2, 3, 4, m = 5\}$ and part B shows the sequence of operations. For example, job 3 is composed of 4 operations. So number 3 is repeated just 4 times in part B. Also job 4 consists of 2 operations, and number 4 is repeated 2 times only.

Step 2. Define the NHGASA parameters

The parameters for GA are as follows: *GN*: generation number (stop criterion); *Pop_size*: population size; *Pc*: cross over rate; *Pm*: mutation rate; *TS*: tournament size; *NE*: number of elite solutions in each generation.

The parameters for SA are as follows: *NS*: number of individuals is improved by SA; *Tr*: temperature in SA; α: cooling rate of SA; *R*: external loop of SA; *S*: internal loop of SA, *NC*: number of no improvement in solutions.

Step 3. Generate random initial population

The initial population is randomly generated in size of Pop_{size}. Since the population is obtained randomly the exploration in the solution space is performed more efficiently.

$$P_t = \{x_1, x_2, \ldots, x_i\}; \quad i = 1, 2, \ldots, Pop_{size}; \quad t = 1, 2, \ldots, GN$$

Step 4. The main genetic algorithm loop

Repeat the following step until $G0 < GN$, $NS0 = 1$, $NC0 = 0$

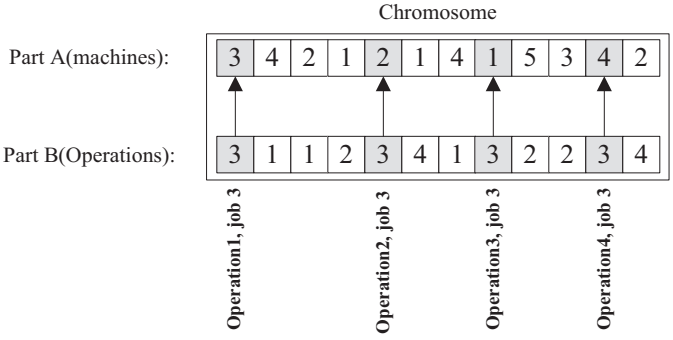

Fig. 2. Representation.

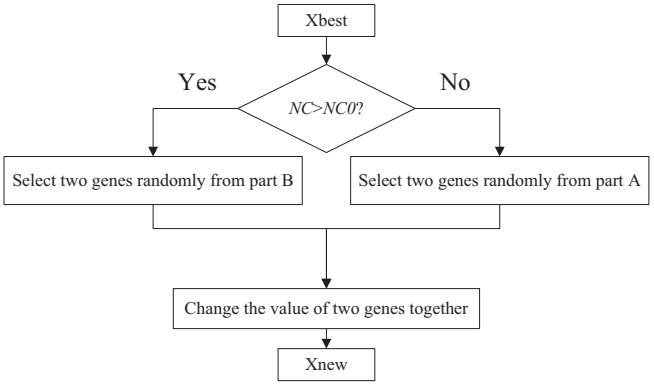

Fig. 3. Altering solutions by SA.

Step 4.1. Calculating three objective functions as follows.

$$F1_{Pt} = \max\{C_{k(i)i,j}\}; \quad K(i) \in K; \quad i \in J; \quad j \in M \tag{13}$$

$$F2_{Pt} = \max\left\{Tj = \sum_{i=1}^{n}\sum_{l=1}^{k(i)} t_{lij}\right\} \tag{14}$$

$$F3_{Pt} = \left\{\sum_{j=1}^{m} T_j\right\} \tag{15}$$

Eq. (13) evaluates makespan and Eq. (14) calculates the workload of the most loaded machine. Eq. (15) calculates the total workload of machines.

M is a set of machines, $M = \{1, 2, \ldots, m\}$ and *J* is a set of jobs $J = \{1, 2, \ldots, n\}$. *K* is a set that shows the number of job's operations. Since $K(i) = L$ means that job *i*th has *L* operations. $C_{k(i)ij}$ is completion time of $K(i)$th operation of job *i* on machine *j*. t_{lij} is processing time of *l*th operation of job *i* on machine *j* and finally T_j is the sum of time that machine *j* is doing process.

Step 4.2. The main loop of SA

Repeat the following steps until $NS > NS0.R0 = 0$, $S0 = 0$

Step 4.2.1. External loop of SA

Select one individual from current population randomly and assign it to the *Xbest*. Evaluate objective functions ($F1_{Xbest}$, $F2_{Xbest}$, $F2_{Xbest}$) by Eqs. (13)–(15). $R0 = R0 + 1$

Repeat following step until $R0 < R$ otherwise go to step (4.2)

Step 4.2.1.1. Internal loop of SA

Repeat following steps while $S0 < S.S0 = S0 + 1$

Alter the current chromosome (*Xbest*) and assign that to the *Xnew*. Then evaluate objective functions ($F1_{Xnew}$, $F2_{Xnew}$, $F2_{Xnew}$) for *Xnew* by Eqs. (13)–(15). The altering method is illustrated in Fig. 3. In this altering method first part B of each chromosome is changed if there is no improvement in all objective values for *NC* times continuously then part A is changed. This approach to altering extremely improves each solution in population.

Then calculate ΔC_i by Eqs. (16)–(18). If Eq. (19) is correct then *Xbest* = *Xnew*.

$$\Delta C_1 = F1_{Xnew} - F1_{Xbest} \tag{16}$$

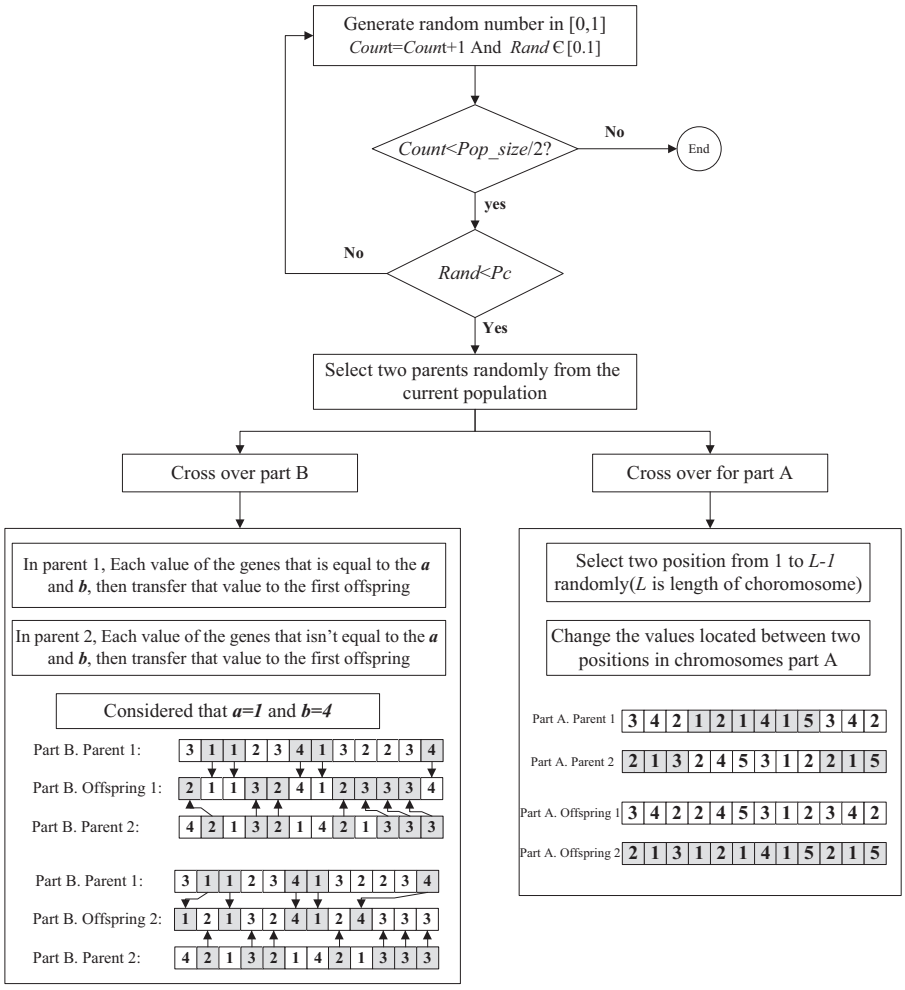

Fig. 4. Crossover operator.

$$\Delta C_2 = F2_{Xnew} - F2_{Xbest} \quad (17)$$

$$\Delta C3 = F3_{Xnew} - F3_{Xbest} \quad (18)$$

$$\text{If } \Delta C_i \le 0; \quad i = 1, 2, 3 \underset{Then}{\longrightarrow} X_{best} = X_{new} \quad (19)$$

If $\Delta C_i \succ 0$ then generate random number $y \in [0,1]$. If Eq. (20) is correct then $Xbest = Xnew$ otherwise $NC0 = NC0 + 1$ and go to step (4.2.1.1).

$$\text{if } \; y < \exp\left(\frac{-\Delta C_i}{T}\right); \quad i = 1, 2, 3; \underset{then}{\longrightarrow} X_{best} = X_{new} \quad (20)$$

Step 4.2.2. $TR = \alpha \times Tr$ and $R0 = R0 + 1$ then go to step (4.2.1) then replace the selected chromosome in step 4.2.1 by *Xbest* and finally $NS0 = NS0 + 1$ and go to step (4.2)

Step 4.3. Main loop of MOGA approach

Calculate ranking of each solution in current population by Eq. (7). Then evaluate distances between all pairs of solutions by Eq. (9). Fitness value for each solution in current population is calculated by Eq. (8). According to the last two steps calculate the *Niche count* for $x \in P$ by Eq. (10) and then calculate the modified fitness value for $x \in P$ by Eq. (11). Finally evaluate the normalized fitness value for $x \in P$ by Eq. (12).

Step 4.4. Elitism approach

Select *NE* elite individuals from current population and move them to the next generation. For this purpose first sort the population according to $fit''(x,t)$. Since all objective functions are minimization, select *NE* optimal individuals in the sorted population and move them to the next population.

Step 4.5. Selection by tournament

Select (*Pop_size_NE*) individuals from current population by tournament selection procedure. First select *TS* individuals randomly and then each of these individuals has optimal $fit''(x,t)$ select and move to the next generation. This process is repeated until the number of solutions in the next generation is equal to *Pop_size*.

Step 4.6. Genetic operators

The genetic operators used in this paper are captured from [21].

Step 4.6.1. Cross over

In this method, two different approaches are proposed for parts A and B. Fig. 4 illustrates the cross over operator.

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015

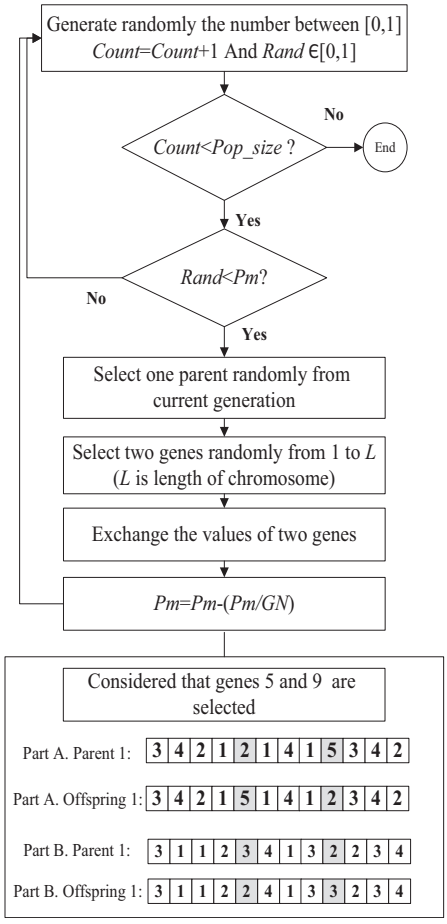

Fig. 5. Mutation operator.

Step 4.6.2. Mutation

Fig. 5 shows the mutation that use in this paper.

Step 4.7. $G0 = G0 + 1$ and go to step (4)

Step 5. Submitting final population as the results

After the following steps are finished, the last population and its objective values are shown as the results of problem.

The *NHGASA* algorithm is explained comprehensively. Now in the next section we prove the strength of the *NHGASA* and compare this method to three other methods. It shows that *NHGASA* has extreme strength to solve FJSP with higher quality and lower computation time in comparison with others.

4. Computational results

In order to evaluate the performance of *NHGASA* algorithm the following popular benchmark is used [27,28]. Two instances of 4-job and 5-machine and 10-job and 10-machine FJSP are presented in Tables 1 and 2. These examples are solved by *NHGASA* algorithm and then the results are compared with three other methods [28–30].

The parameters that are defined for *NHGASA* algorithm are as follows:

4-job and 5-machine parameters: $GN = 100$, $Pop\_size = 100$, $Pc = 0.9$, $Pm = 0.1$, $TS = 10$, $Tr = 10$, $\alpha = 0.1$, $R = 20$, $S = 5$, $NS = 20$, $NE = 10$

Table 2
10 × 10 with 30 operations problem.

Job	Operation	Processing time for machine *Mi*									
		M1	*M2*	*M3*	*M4*	*M5*	*M6*	*M7*	*M8*	*M9*	*M10*
J1	1	1	4	6	9	3	5	2	8	9	5
	2	4	1	1	3	4	8	10	4	11	4
	3	3	2	5	1	5	6	9	5	10	3
J2	1	2	10	4	5	9	8	4	15	8	4
	2	4	8	7	1	9	6	1	10	7	1
	3	6	11	2	7	5	3	5	14	9	2
J3	1	8	5	8	9	4	3	5	3	8	1
	2	9	3	6	1	2	6	4	1	7	2
	3	7	1	8	5	4	9	1	2	3	4
J4	1	5	10	6	4	9	5	1	7	1	6
	2	4	2	3	8	7	4	6	9	8	4
	3	7	3	12	1	6	5	8	3	5	2
J5	1	7	10	4	5	6	3	5	15	2	6
	2	5	6	3	9	8	2	8	6	1	7
	3	6	1	4	1	10	4	3	11	13	9
J6	1	8	9	10	8	4	2	7	8	3	10
	2	7	3	12	5	4	3	6	9	2	15
	3	4	7	3	6	3	4	1	5	1	11
J7	1	1	7	8	3	4	9	4	13	10	7
	2	3	8	1	2	3	6	11	2	13	3
	3	5	4	2	1	2	1	8	14	5	7
J8	1	5	7	11	3	2	9	8	5	12	8
	2	8	3	10	7	5	13	4	6	8	4
	3	6	2	13	5	4	3	5	7	9	5
J9	1	3	9	1	3	8	1	6	7	5	4
	2	4	6	2	5	7	3	1	9	6	7
	3	8	5	4	8	6	1	2	3	10	12
J10	1	4	3	1	6	7	1	2	6	20	6
	2	3	1	8	1	9	4	1	4	17	15
	3	9	2	4	2	3	5	2	4	10	23

Table 3
Pareto optimal solutions of 4 × 5 problem by NHGASA algorithm.

Objective function	*F1*	*F2*	*F3*
Pareto front	11	10	32
	13	7	33
	12	8	32
	11	9	34

10-job and 10-machine parameters: $GN = 200$, $Pop\_size = 100$, $Pc = 0.9$, $Pm = 0.1$, $TS = 10$, $Tr = 10$, $\alpha = 0.1$, $R = 20$, $S = 5$, $NS = 40$, $NE = 10$

The *NHGASA* model is programmed by Visual Basic Application (VBA) and running on a 2 GHz PC with 512 MB RAM. Then all the parameters that are defined above and Tables 1 and 2 are introduced as inputs to the program. The outputs of the program are illustrated in Tables 3 and 4. Since *NHGASA* algorithm uses Pareto optimal solutions, in the last generation the set of optimal solutions are calculated as the results.

Table 4
Pareto optimal solutions of 10 × 10 problem by NHGASA algorithm.

Objective function	*F1*	*F2*	*F3*
Pareto front	8	7	41
	8	5	42
	7	6	42
	7	5	43

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015

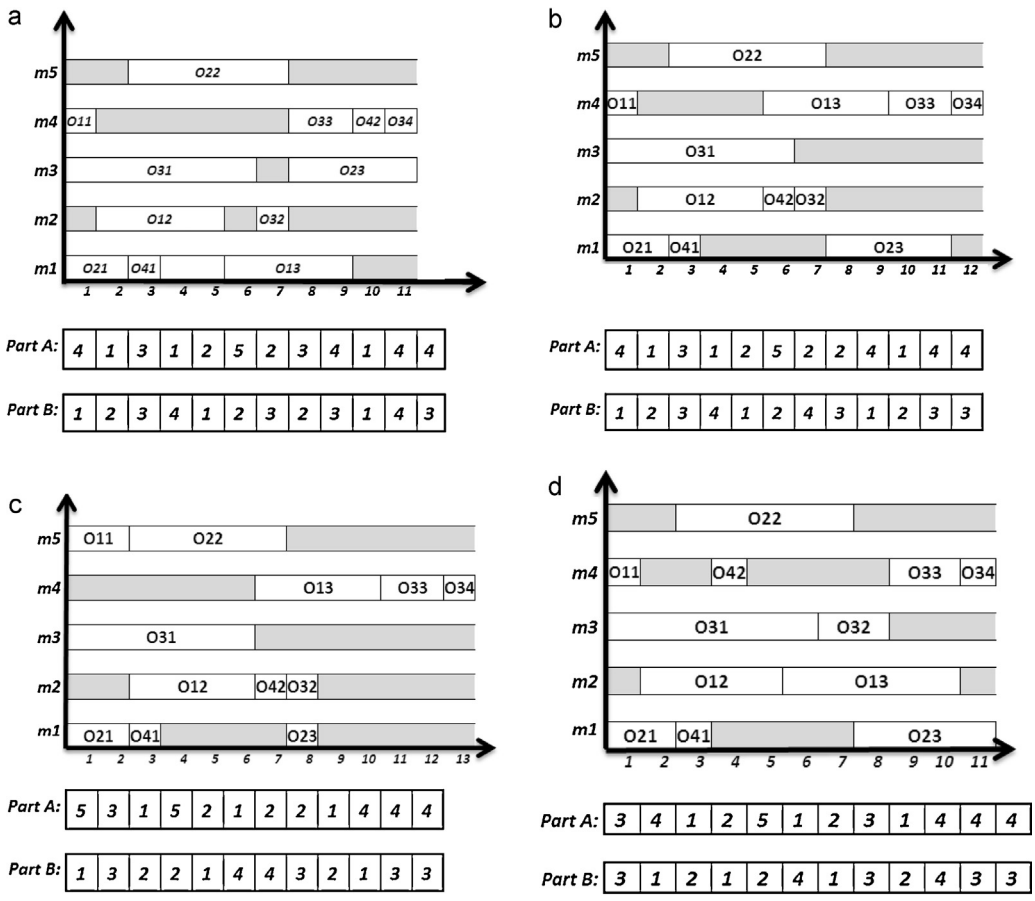

Fig. 6. (a) The Gantt chart of Pareto optimal solution 1 (problem 4 × 5). (b) The Gantt chart of Pareto optimal solution 2 (problem 4 × 5). (c) The Gantt chart of Pareto optimal solution 3 (problem 4 × 5). (d) The Gantt chart of Pareto optimal solution 4 (problem 4 × 5).

Table 5
Comparison of the NHGASA and the AL-CGA (4 × 5 problem).

Algorithm	AL-CGA			*NHGASA*			*NHGASA*		
Objective function	*F*1	*F*2	*F*3	*F*1	*F*2	*F*3	Improvement in *F*1	Improvement in *F*2	Improvement in *F*3
Solution 1	18	8	32	12	8	32	6	0	0
Solution 2	18	7	33	13	7	33	5	0	0
Solution 3	16	9	35	11	9	34	5	0	1
Solution 4	16	10	34	11	10	32	5	0	2

Table 6
Comparison of the NHGASA and other algorithms (4 × 5 problem).

Algorithms	*F*1	*F*2	*F*3	NHGASA		
				*F*1	*F*2	*F*3
MOEA-	16	8	32	12	8	32
GLS	16	7	33	13	7	33
GA + HA	13	7	33	11	10	32
	12	8	32	11	9	34

Table 7
Comparison of the NHGASA and other algorithms (10 × 10 problem).

Algorithms	*F*1	*F*2	*F*3	NHGASA		
				*F*1	*F*2	*F*3
AL-CGA	8	7	41	8	7	41
	8	5	42	8	5	42
	7	5	45	7	5	43
PSO-SA	7	6	44	7	6	42
GA-HC	8	5	42			
	7	5	43			

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015

Table 8
Comparison of the NHGASA and other algorithms (10 × 10 problem).

Algorithms	MOEA-GLS			*NHGASA*		
	*F*1	*F*2	*F*3	*F*1	*F*2	*F*3
Solutions	8	7	41	8	7	41
	8	5	42	8	5	42
	7	5	43	7	5	43
	7	6	42	7	6	42
Time	19.77 s			18.44 s		

The most important and influential aspect of *NHGASA* compared to other algorithms that makes it more efficient is that in the last generation all solutions in Tables 3 and 4 are calculated and illustrated simultaneously. In other words, all Pareto optimal solutions are calculated simultaneously and there is no need to run program more than once. Figs. 6a–d and 7a–d show the schedule of its relevant chromosomes.

Fig. 8a–d shows the evolutionary trend of each population according to the normalized fitness value ($fit''(x,t)$) in the tth generation for 4-job and 5-machine problem. Different values for t are 30, 60 and 100. As shown in Fig. 8a, initial population is scattered into the whole solution space. This issue causes better exploration in solution space. Fig. 8c illustrates convergence of solutions to the Pareto optimal solutions. Fig. 8d shows Pareto optimal solutions in the last generation. As it has been seen in this figure, four solutions are evaluated simultaneously.

4.1. Problem 4-jobs and 5-machines (4 × 5)

Now *NHGASA* is compared with three other algorithms. In the [28] the 4-job and 5-machine FJSP (Table 1) is solved by AL-CGA method. The results of *NHGASA* and AL-CGA algorithm are shown in Table 5.

As shown in Table 5 *NHGASA* algorithm is immensely stronger than AL-CGA in the first objective function (*F*1). While in the other two objective functions *NHGASA* is equal or better than AL-CGA method.

Also this method is compared with GA-HC introduced in [29] and MOEA-GLS that proposed in [30]. Table 6 illustrates this comparison.

4.2. Problem 10-jobs and 10-machines (10 × 10)

In order to evaluate the performance of proposed algorithm in middle-size problems, 10-job and 10-machine FJSP (Table 2) is solved by *NHGASA* and compared with AL-CGA, MOEA-GLS, GA-HC and also PSO-SA [19]. The results are shown in Tables 7 and 8.

As shown in Table 7 *NHGASA* obtains four solutions as Pareto front for solving 10-job and 10-machine FJSP and the quality of *NHGASA* solutions are higher than PSO-SA and also AL-CGA algorithms. Although the *NHGASA* and MOEA-GLS algorithm obtain the same Pareto front to solve 10 × 10 FJSP, the computation time for *NHGASA*, as illustrated in Table 8, is lower than MOEA-GLS which shows the *NHGASA* higher performance.

All solutions that are obtained from *NHGASA* have two important and significant characteristics:

1. *NHGASA* obtains better solution than other methods.
2. This algorithm is capable of calculating the best solutions of other methods merely by running the algorithm once.

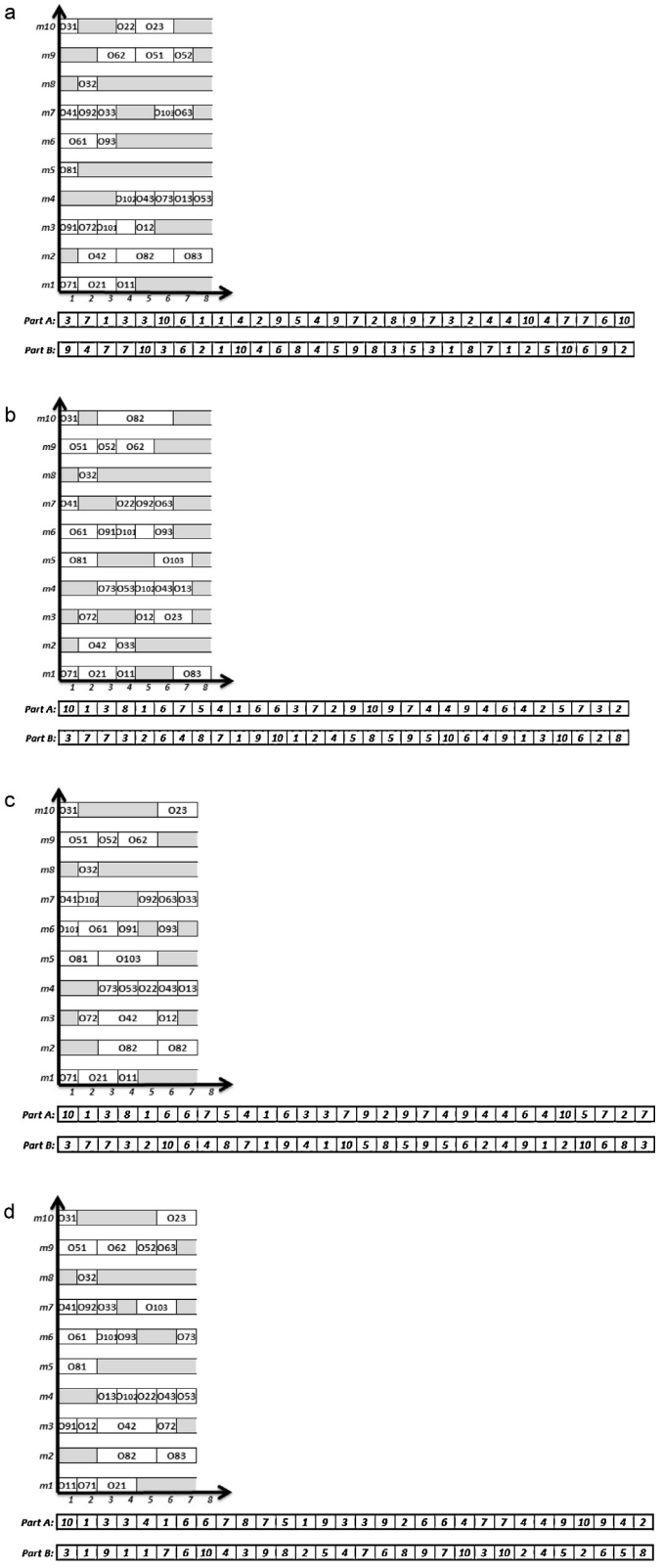

Fig. 7. (a) The Gantt chart of Pareto optimal solution 1 (problem 10 × 10). (b) The Gantt chart of Pareto optimal solution 2 (problem 10 × 10). (c) The Gantt chart of Pareto optimal solution 3 (problem 10 × 10). (d) The Gantt chart of Pareto optimal solution 4 (problem 10 × 10).

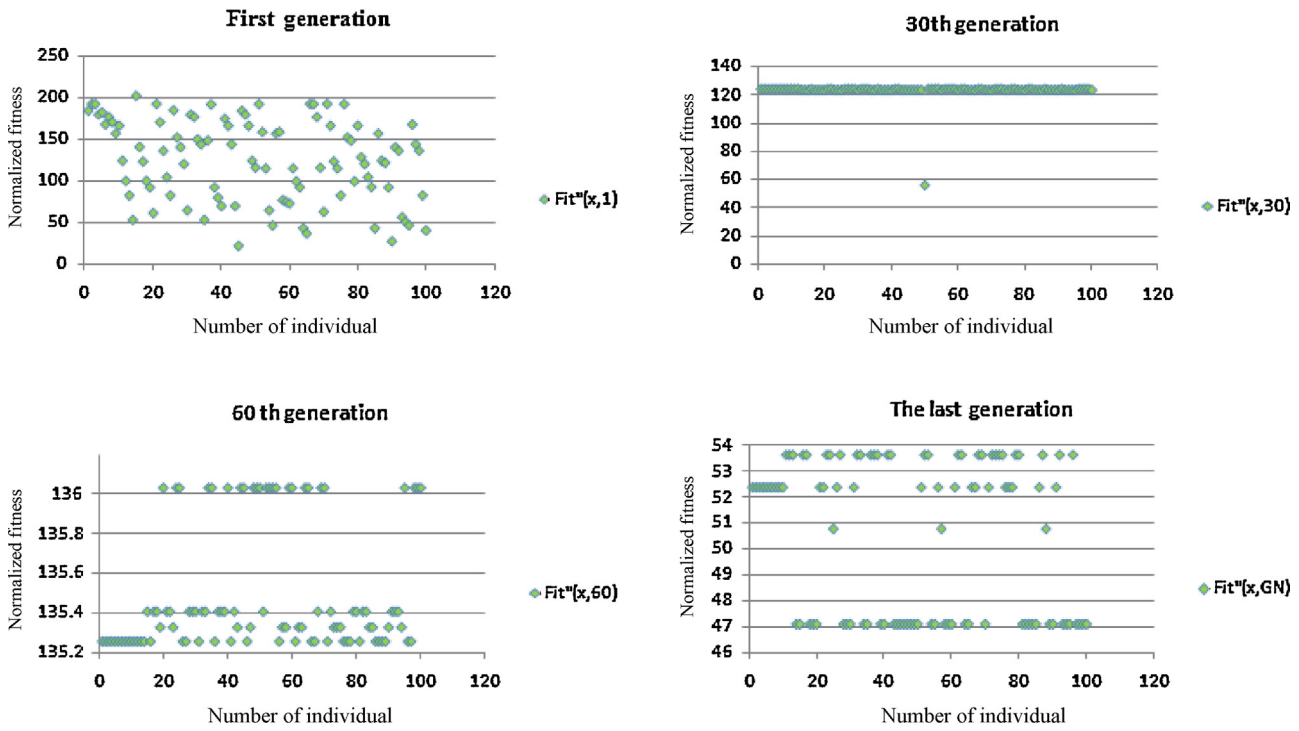

Fig. 8. Evolution of solutions in each generation.

5. Conclusion and future works

In this paper, a new method called *NHGASA*, has been proposed for one of the hardest combinatorial optimization FJSP. In the *NHGASA* algorithm, Pareto approaches are used to solve FJSP. By using Pareto optimal solutions the set of optimal solutions is obtained instead of an individual optimal solution. The high performance of *NHGASA* undertakes obtaining better solutions or the same best solutions of previous works. In the previous sections, *NHGASA* is compared with other methods. The quality of solutions in the *NHGASA* is immensely higher than those of others. While MOGA has slow convergence to the optimal solution, it has proposed and added elitism approaches to the *NHGASA* to decline the computation time. Also applying modified approach for solutions by the simulated annealing algorithm has caused higher quality of solutions.

More comprehensive studies can be conducted to apply the *NHGASA* for other combinatorial optimization problems. Different FJSP data with larger sizes will be gathered and used to solve FJSP with *NHGASA*.

References

[1] Garey MR, Johnson DS, Sethi R. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research 1976;1:117–29.
[2] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job shop scheduling problem. Computers & Operations Research 2008;35:3202–12.
[3] Zhang G, Gao L, Shi Y. An effective genetic algorithm for the flexible job shop scheduling problem. Expert Systems with Applications 2011;38:3563–73.
[4] Brucker P, Schile R. Job shop scheduling with multi-purpose machines. Computing 1990;45:369–75.
[5] Thamilselvan R, Balasubramanie P. Integrating genetic algorithm tabu search approach for job shop scheduling. International Journal of Computer Science and Information Security 2009;2:1–6.
[6] Najid NM, Dauzere-Peres S, Zaidat A. A modified simulated annealing method for flexible job shop scheduling problem. In: IEEE international conference on system, man and cybernetics, vol. 5. 2002.
[7] Colorni A, Dorigo M, Maniezzo V, Trubian M. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 2002;34:39–54.
[8] Chen H, Ihlow J, Lehmann C. A genetic algorithm for flexible job-shop scheduling. In: Proceeding in IEEE international conference on robotics and automation, vol. 2. 1999. p. 1120–5.
[9] Li B, Wu S, Yang J, Du M. A three-fold approach for job shop problems: a divide-and-integrate strategy with immune algorithm. Journal of Manufacturing Systems 2012;31:195–203.
[10] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 1993;41:157–83.
[11] Hurink J, Jurisch B, Thole M. Tabu search for the job shop scheduling problem with multi-purpose machines. OR Spectrum 1994;15:205–15.
[12] Dauzère-Pérès S, Paulli J. An integrated approach for modeling and solving the general multi-processor job-shop scheduling problem using tabu search. Annals of Operations Research 1997;70:281–306.
[13] Yang JB. GA-based discrete dynamic programming approach for scheduling in FMS environments. IEEE Transactions on Systems, Man and Cybernetics, Part B 2001;31:824–35.
[14] Zhang HP, Gen M. Multistage-based genetic algorithm for flexible job shop scheduling problem. Journal of Complexity International 2005;48:409–25.
[15] Tay JX, Ho NB. Evolving dispatching rules using genetic programming for solving multi-objective flexible job shop problems. Computers & Industrial Engineering 2008;54:453–73.
[16] Wang L, Wang S, Xu Y, Zhou G, Liu M. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Computers & Industrial Engineering 2011;64:917–26.
[17] Zhang Q, Manier H, Manier MA. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computer & Operations Research 2011;39:1713–23.
[18] Ulungu EL, Teghem J, Fortemps PH, Tuyttens D. MOSA method: a tool for solving MOCO problems. Journal of Multi-Criteria Decision Analysis 1999;8:221–36.
[19] Xia W, Wu Z. An effective hybrid optimization approach for multi-objective flexible job shop scheduling problems. Computer & Industrial Engineering 2005;48:409–25.
[20] Zhang GH, Shao XY, Li PG, Gao L. A genetic algorithm and tabu search for multi objective flexible job shop scheduling problems. IEEE International Conference on Computing, Control and Industrial Engineering 2010;1:251–4.
[21] Zhang GH, Shao XY, Li PG, Gao L. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job shop scheduling problem. Computer & Industrial Engineering 2009;56:1309–18.
[22] Bagheri A, Zandieh M. Bi-criteria flexible job shop scheduling with sequence-dependent setup times—variable neighborhood search approach. Journal of Manufacturing Systems 2011;30:8–15.
[23] Jones DF, Mirrazavi SK, Tamiz M. Multi objective meta-heuristics: an overview of the current state-of-the-art. European Journal of Operational Research 2002;137:1–9.

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015

[24] Konak A, Coit DW, Smith AE. Multi objective optimization using genetic algorithms: a tutorial. Reliability Engineering & System Safety 2006;91:992–1007.
[25] Fonseca CM, Fleming PJ. Multi objective genetic algorithms. In: IEE colloquium on genetic algorithm for control systems engineering (digest no. 1993/130). 1993.
[26] Fonseca CM, Fleming PJ. Genetic algorithms for multi objective optimization: formulation, discussion and generalization. In: Proceeding of the ICGA-93: fifth international conference on genetic algorithm. 1993. p. 416–23.
[27] Kacem I, Hammadi S, Borne P. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computers in Simulation 2002;60:245–76.
[28] Kacem I, Hammadi S, Borne P. Approach by localization and multi objective evolutionary optimization for flexible job shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C 2002;32:1–13.
[29] Motaghedi A, Sabari K, Heydari M. Solving flexible job shop scheduling with multi objective approach. International Journal of Industrial Engineering & Production Research 2010;21:197–209.
[30] Ho NB, Tay JC. Solving multiple-objective flexible job shop problems by evolution and local search. IEEE Transactions on Systems Man, and Cybernetics, Part C 2008;38:674–85.

Please cite this article in press as: Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst (2013), http://dx.doi.org/10.1016/j.jmsy.2013.04.015