Decision Science Letters 3 (2014) 157–168

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A Pareto archive floating search procedure for solving multi-objective flexible job shop scheduling problem

J. S. Sadaghiani[a], Soheil Azizi Boroujerdi[a*], Mohammad Mirhabibi[a] and P. S. Sadaghiani[b]

[a]*Department of Management, Allame Tabatabaei University, Tehran, Iran*
[b]*Department of Management, Qazvin Azad University, Qazvin, Iran*

C H R O N I C L E

Article history:
Received October 23,2013
Received in revised format
November 25, 2013
Accepted December 12, 2013
Available online
December 14 2013

Keywords:
Flexible job shop scheduling
Variability
Multi-objective Meta-heuristic algorithm
Floating search procedure

A B S T R A C T

Flexible job shop scheduling problem is a key factor of using efficiently in production systems. This paper attempts to simultaneously optimize three objectives including minimization of the make span, total workload and maximum workload of jobs. Since the multi objective flexible job shop scheduling problem is strongly NP-Hard, an integrated heuristic approach has been used to solve it. The proposed approach was based on a floating search procedure that has used some heuristic algorithms. Within floating search procedure utilize local heuristic algorithms; it makes the considered problem into two sections including assigning and sequencing sub problem. First of all search is done upon assignment space achieving an acceptable solution and then search would continue on sequencing space based on a heuristic algorithm. This paper has used a multi-objective approach for producing Pareto solution. Thus proposed approach was adapted on NSGA II algorithm and evaluated Pareto-archives. The elements and parameters of the proposed algorithms were adjusted upon preliminary experiments. Finally, computational results were used to analyze efficiency of the proposed algorithm and this results showed that the proposed algorithm capable to produce efficient solutions.

1. Introduction

Job shop production systems are considered as one of the most common forms of production systems in production systems. Therefore, there have been extensive efforts to increase the efficiency of these systems. There is also an increasing intensity of competition among manufacturing firms to reduce prices, timely delivery and customer satisfaction led to efficient use of resources and to increase productivity per unit of production. Strong planning in production systems plays essential role in increasing productivity and customer satisfaction. Production scheduling is the problem, which effects on timely delivery and efficient use of organization's resource. In a scheduling problem, start and finish time of tasks, machines sequences etc. are determined. Although many researchers have

* Corresponding author.
E-mail addresses: azizi912@atu.ac.ir, Mirhabibi912@atu.ac.ir (S. Azizi Boroujerdi)

doi: 10.5267/j.dsl.2013.12.001

proposed various scheduling models in the past, there is still a need for a comprehensive model to address the needs of organizations. Job shop scheduling is one of the production scheduling problems and is one of the most important mix optimization problems.

The n×m classical JSP is an NP-hard problem (Garey et al., 1976) that involves *n* jobs and *m* machines. Each job is to be processed on each machine in a predefined sequence and each machine processes only one job at a time. In practice, the shop-floor setup typically consists of multiple copies of the most critical machines so that bottlenecks due to long operations or busy machines could be reduced. As such, an operation may be processed on more than one machine having the same function. This leads to a more complex problem known as the flexible job-shop scheduling problem (FJSP). The extension involves two tasks: assignment of an operation to an appropriate machine and sequencing the operations on each machine. In addition, for complex manufacturing systems, a job can typically visit a machine more than once, which called as recirculation. These three features of the FJSP significantly increase the complexity of finding even approximately optimal solutions (Garey et al., 1976). Furthermore, instead of considering only a single objective, most scheduling problems in practice involve simultaneous optimization of various competing objectives. Therefore, in order to tackle the FJSP problems found in practice, efficient optimization strategies are applied to deal with both multiple objectives and exponential search space complexity.

In recent years, for solving the JSP and FJSP many meta- heuristic algorithms such as Tabu-search, particle swarm optimization, Ant colony optimization and genetic algorithms are proposed. For solving FJSP, different hierarchical and integrated approaches have been implemented. In hierarchical approach, the problem is divided into two sub-assignment and sequencing problems and they are solved, independently. Pauli (1995) as well as Saidi-Mehrabad and Fattahi (2007) extended this approach to solve FJSP. However, in the integrated approach, assignment and sequencing are considered, simultaneously. Hurink et al. (1994) and Dauzère-Pérès and Paulli (1997) developed tabu-search algorithm to solve FJSP with integrated approach.

In addition, two approaches are used for solving FJSP problem and type of solution structure is an important role in solving process. We could mention to some of solution seed structures used for solving FJSP. Mesghouni et al. (1997) proposed the Parallel Jobs Representation in a form of a matrix where each row represents a job and each entry is a pair value. An indirect representation containing a pair of chromosomes, A and B, was proposed by Chen et al. (1997) where A is a string of machine assignments and B is a string of sequencing. Gen et al. (2009) used a multi-stage operation-based GA (MOGA) to simplify the chromosome. A MOGA chromosome is basically a routing string, one locus for each operation. The Assignment Table is proposed by Kacem et al. (2002). The assignment table is an Op (total number of operations) × M table. Each row represents an operation.

Gen et al. (2009) represented an assignment Table where each member dedicated the operation and included machine number, start and finish time of operation. Fattahi et al. (2007) proposed a method, which involves two matrices. The first matrix introduces assignment and the second matrix introduces operation's sequence to machines. Gao et al. (2007) also used a pair of chromosomes, where the machine assignment string was a fixed assignment similar to the A-string proposed by Chen et al. (1997), with permutations with repetition used for scheduling. Ho et al. (2007) presented two-part structure for assignment and sequencing. Based on the dispatching rules, Unachak and Adviser-Goodman (2010) presented a two-part structure of strategies for assignment and sequencing.
In recent years, development of FJSP's goal function from single-objective to multi-objective has been mentioned and it was investigated in both cumulative weighting and Pareto approaches. In the first method, each objective has a weight and problem was converted to single objective function, which can be solved by using single objective meta-heuristics algorithms.

Pareto set approach provided non-dominated solutions for decision maker by using multi objective Meta heuristics algorithm. Kacem et al. (2002) presented a hybrid approach based on fuzzy logic and multi-objective evolutionary algorithm to the problems of flexible job shop scheduling. They provided three objectives: minimization of make span, total workload and maximum workload of jobs in the models and prepare an appropriate solution seed structure for solving it. Baykasoğlu and Sönmez (2004) presented a tabu-search multi objective algorithm. Xia and Wu (2005) treated this problem with a hybrid of particle swarm optimization and simulated annealing as a local search algorithm. Tay and Ho (2008) proposed CDR algorithm, showed that CDR algorithm could strikingly enhance the quality of production scheduling and used it for scheduling FJSP and took brilliant results. Hierarchical approach reduces the complexity of the problem by making the search space smaller. The integrated approach considers both sub problems together and search randomly in two spaces. After finding an appropriate assignment solution, a job shop problem is solved in hierarchical mode. Therefore, some good solutions have driven out because of a job shop problem has solved for specific assignment whereas changing the assignment may provide better solution. In an integrated mode because there is no particular trend except randomly variations between assignment space the sequence, this pattern of searching tends to increase variability in the search space. Thus, it is essential to design a trend as well as to decrease complexity of problem by performing a complete search in the search space.

In this paper, a meta-heuristic algorithm and a heuristic local search algorithm are combined, and a new approach presented. This method reduces variability in search space and it has the benefits of both of hierarchical and integrated approaches. The structure of this paper is organized as follows. In section 2, assumptions, constraints and objectives of FJSP will be illustrated. In Section 3, Floating search approach and NSGAII algorithm will be introduced for solving MOFJSP. The proposed algorithm will be presented in Section 4. Computational results, analysis and discussion will be given in Section 5. Finally, Section 6 Conclusion and suggestions will be presented for future researches.

2. Problem description

A N×M Flexible Job-shop Scheduling Problem consists of N jobs and M machines. Each job $J = 1, \ldots, N$ has asequence of operations $O_{j,h}, h = 1,\ldots,h_j$, that $O_{j,h}$ and h_j in that order represented the h operation of job j and the number of operation of job j that is required. The set of machines includes $M = \{m_1, m_2, \ldots, m\}$. Index j indicates the job h indicates the operation and indicates the machine. The operation $O_{j,h}$ can process on machine $M_{j,h} \subset M$ with processing time $P_{i,j,h}$. Set $M_{j,h}$ is defined on the base of $a_{i,j,h}$ and index *k* is identified a set of operations assigned to each machine. According to the description, parameters of the model are:

n: number of jobs

m: number of machines

$a_{i,j,h}$: represent the operations assigned to machines, which are defined as follows.

$$a_{i,j,h} = \begin{cases} 1 \text{ if } O_{j,h} \text{ is done on machine i} \\ 0 \text{ elsewhere} \end{cases}$$

$P_{i,j,h}$: processing time $O_{j,h}$ if the machine i process on it,

The decision variables are:

$C_{\max}$: Make span

w_i : Workload of machine i

$$y_{i,j,h} = \begin{cases} 1 \text{ if machine i is selected for } O_{j,h} \\ 0 \quad \text{elsewhere} \end{cases}$$

$$x_{i,j,h,k} = \begin{cases} 1 \text{ if } O_{j,h} \text{ is done on machine i in priority k} \\ 0 \quad \text{elsewhere} \end{cases}$$

$t_{j,h}$: Starting time of operation $O_{j,h}$

$Tm_{i,k}$: Start time machine i is used in priority k

K_i : The number of operations assigned to machine i

$Ps_{j,h}$: processing time $O_{j,h}$ after choosing the machine for its processing

According to the items listed above, the linear programming model is presented below:

$$\min (f_1 = C_{\max}, f_2 = \max_{1 \le i \le m} w_i \; f_3 = \sum_{i=1}^{m} w_i)$$

subject to

$$c_{\max} \ge t_{j,h_j} + Ps_{j,h_j} \quad for\ j = 1,\ldots,n; \tag{1}$$

$$\sum_i y_{i,j,h} . P_{i,j,h} = Ps_{j,h} \; for\ j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{2}$$

$$t_{j,h} + Ps_{j,h} \le t_{j,h+1} \; for\ j = 1,\ldots,n; h = 1,\ldots,h_{j-1}; \tag{3}$$

$$Tm_{j,k} + Ps_{j,h} . x_{i,j,h,k} \le Tm_{j,k+1} \; for\ i = 1,\ldots,m; \;\; j = 1,\ldots,n; h = 1,\ldots,h_j; k = 1,\ldots,k_j - 1; \tag{4}$$

$$Tm_{j,k} \le t_{j,h} + (1 - x_{i,j,h,k}).L \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots,h_j; k = 1,\ldots,k_i; \tag{5}$$

$$Tm_{i,k} + (1 - x_{i,j,h,k}).L \ge t_{j,h} \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots,h_j; k = 1,\ldots,k_i; \tag{6}$$

$$y_{i,j,h} \le a_{i,j,h} \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{7}$$

$$\sum_j \sum_h x_{i,j,h,k} = 1 \; for\ i = 1,\ldots,m;; k = 1,\ldots,k_i; \tag{8}$$

$$\sum_i y_{i,j,k} = 1 \; for\ j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{9}$$

$$\sum_k x_{i,j,h,k} = y_{i,j,h} \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{10}$$

$$t_{j,h} \ge 0 \; for\ j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{11}$$

$$Ps_{j,h} \ge 0 \; for\ j = 1,\ldots,n; h = 1,\ldots,h_j; \tag{12}$$

$$Tm_{i,k} \ge 0 \; for\ i = 1,\ldots,m; k = 1,\ldots,k_i; \tag{13}$$

$$x_{i,j,h,k} \in \{0,1\} \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots h_j; k = 1,\ldots,k_i; \tag{14}$$

$$y_{i,j,h} \in \{0,1\} \; for\ i = 1,\ldots,m; j = 1,\ldots,n; h = 1,\ldots,h_j. \tag{15}$$

The objective function f_1 determines make span, objective function $f_2$determines maximum workload and finally, the objective function f3 determine total workloads of machines. Constraint (1) is equal to the make span. Constraint (2) determines the process time of operation $O_{j,h}$ on the selected machine. Constraint (3) limits the predecessor of the operations. Constraints (4), (5) and (6) are associated with operation process when the specify machine is free and its predecessor is processed. Constraint (7) specifies that the machines can process each operation. A constraint (8) assigns the operations to the machine and specifies sequence of operations on each machine. Constraints (9) and (10) specify that each operation is processed only in one priority and on one machine.

3. NSGA II algorithm

NSGA-II is an elitist multi-objective evolutionary algorithm, which carries out an approximation of the Pareto front, based on the non-dominance concept. For achieving different Pareto fronts, a ranking procedure is performed at each generation. Now this algorithm is one of the efficient ways to solve multi-objective problems. NSGAII is also noticed in some multi-objective flexible job shop problems and researchers such as Moradi et al. (2011) and Frutos et al. (2010) have already used it.
First, a random initial population P_0 of size N is generated.This population is sorted based on the non-domination aspect. In multi-objective optimization, if two objective functions f1 andf2 are to be minimized then, for any two decision vector *x* and *y*, it is said that *x* dominates if (f1(x) ≤ f1(y) and f2(x) < f2(y)) or (f1(x) < f1(y) and f2(x) ≤ f2(y)). The non-dominated solutions are those that other solutions do not dominate them. Besides, a set of non-dominated solution, achieved by an evolutionary algorithm is called Pareto front. After initialing the P_0, a non-domination level (1 is the best level) is employed to evaluate the solutions. Then, child population Q_0 of size N is created using tournament selection, crossover and mutation operators. For a generation $t \geq 1$, the process is totally different. At the first phase, the population $R_t = Pt \cup Qt$ of size 2N is produced and a non-dominated sortingprocedure (ranking) is applied to return the list of non-dominated fronts. The second phase is dedicated to generation of a new parent population P_{t+1} contained the *N* best solutions. Completing the P_{t+1} with the remaining $N - |P_{t+1}|$ solutions, the crowding procedure is utilizedto the first front not included. Finally, a new child population Q_{t+1} of size *N* are created applying Selection, crossover and mutation operators on the population P_{t+1}.

NSGA-II Algorithm:

Step 0: Produce initial population (P_0) randomly in a size of *N* and then use famous genetic selection to produce children and children population (q_0) are produced.

Step1: Population of children and parents are combined, and (R_t) is built: $R_t = P_t \cup Q_t$

Step2: Set $i = 1, P_t + 1 = \phi$, and then until $|P_{t+1}| + |F_i| < N$ repeat the following operations: $\{P_{t+1} = P_{t+1} \cup F_i, i = i + 1\}$

Step 3: Run Crowding distance procedure for the ranked solutions in (F_i) sets and complete set (P_{t+1}) for $(N - |P_{t+1}|)$ remained solutions

Step 4: Create the population of children (Q_{t+1}) by using the Crowding distance procedure and also the operators of crossover and mutation.

The Crowding distance procedure

Step 1: put $L = |F|$ or L equal to size of F set, for each solution i from F set, we put $d_i = 0$.

Step 2: for each objective function $i = 1,2,3$ sorting f_i set by descending order and in term of their value. The index sort vector I^i that $I^i = Sort(f_i >)$ is produced in it.

Step 3: from $m = 1$ to $m = M$ dedicate a large amount for limitations of solutions or $d_{I_1^m} = d_{I_2^m} = \infty$ and for all other solutions $j = 2,3,...,(l-1)$ put: $d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{\max} - f_m^{\min}}$

Non-dominated sorting procedure:

Step 1: For $p,q \in P$ (P current population) if p dominated q, put: $S_p = S_q + \{q\}$.

Ifq dominated p, put: $n_p = n_q + 1$

If $n_p = ()$ put $F_1 = F_1 \cup \{p\}$

Step2: put i=1, until $F_1 = \Phi$

Put $H_1 = \Phi$

For each $p \in F_i$ and each $q \in s_p$: $n_q = n_q - 1$

If $n_p = ()$ put: $H = H \cup \{q\}$

$i = i + 1$ $F_1 = H$ (Deb, 2001)

In order to generate an initial population, random generation and heuristic's algorithms (Ho et al., 2007) is used. In this method, for each random permutations of job, a feasible solution is produced by using dispatching rules.

3.1. Crossover and mutation algorithms

Heuristics methods are used for matching floating search approach with NSGAII observed in Fig 1. In these methods, two heuristics local search algorithms, sequence and assignment neighborhood search algorithms, were designed to set crossover based on the critical path method. The critical path search is a procedure that does an effective neighborhood search around the detected feasible solution. Critical path method was introduced by Adams et al. (1988) in JSP. A critical path begins from the operation that has a finishing time equal to make span and it continues each time on the operations where their finishing time is equal to the operation starting time, which exist in the critical path to be first. This algorithm is based on the following steps:

- ✓ Produce random number $r \in (0,1)$.
- ✓ If $r \prec 0.5$ then run the crossover operation assignment part.
- ✓ Run the neighborhood search algorithm for sequence. If the solution is improved then replace with First solution, otherwise the first solution is remained.
- ✓ If $r \succ 0.5$ then run the crossover operator on the sequence part.
- ✓ Run the neighborhood search algorithm for assignment. If the solution is improved then replace with First solution, otherwise the first solution is remained.

3.2 Neighborhood search algorithms for sequence

Step 1: determine Critical path for the current solution and order operations to public critical , critical and non-critical. Public critical operations are in two or more critical paths.

Step 2: Make a list of capable machines for all operations of jobs and specify the allocated machine for each one**.**

Step 3: For the critical operations that exist in the list choose the machine with least process time (if more than one choose the least workload) and replace. If there wasn't such a machine, the next operation in the list is considered**.**

Step 4: If the entire list were reviewed and there was no change, please stop.

3.3 Neighborhood search algorithm to assignment

Step 1: Find the critical paths for the current solution,

Step 2: Arrange machines on the base of building period in descending order,

Step 3: choose the first machine in the list and two critical operations, which are not belong to a same job are swapped. If it's not possible, check the next machine in the list.

Step 4: Finish

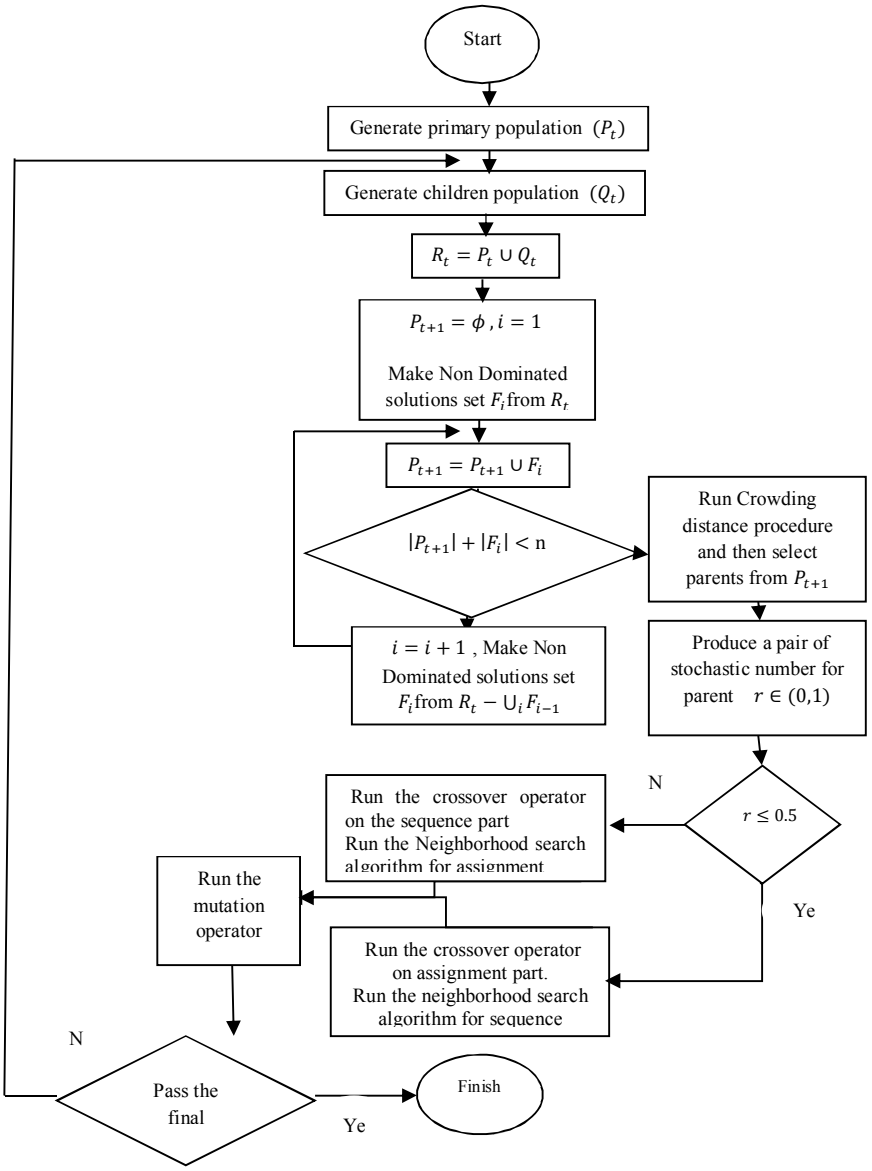

Fig. 1. Structure of the proposed algorithm

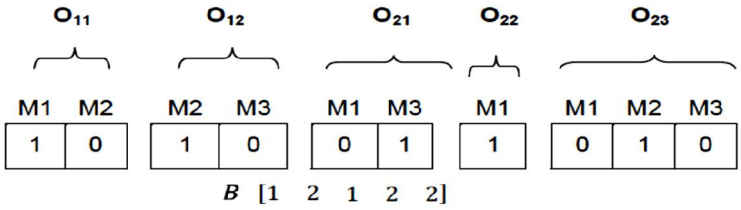

Fig. 2. Encoding the structure presented by Ho et al. (2007)

For doing crossover on the assignment part, two point crossovers were used. Crossover on the sequence part is in the following steps:

- ✓ one job is randomly chosen and all operations of it are removed from parent 2
- ✓ Put the remain operations from parent 2 in empty places in a way that their place don't change
- ✓ Repeat step 1 and 2 for the parent 1.

In Fig. 3(a) and Fig. 3(b) crossover operators of the assignment and sequence are shown on an example.

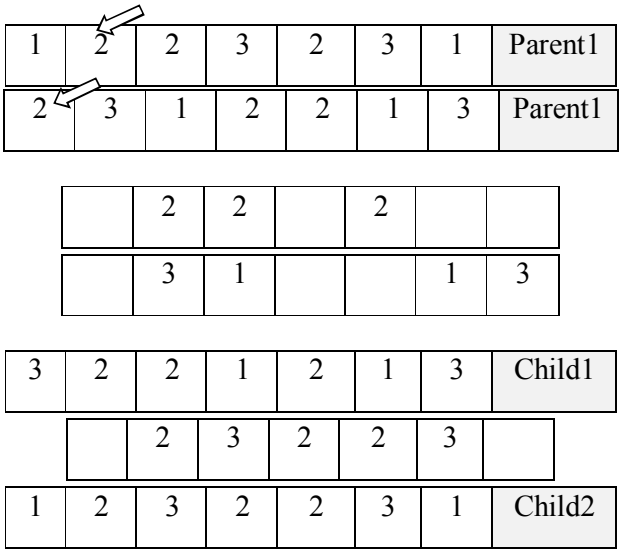

Fig. 3(a). Example of crossover on the sequence part

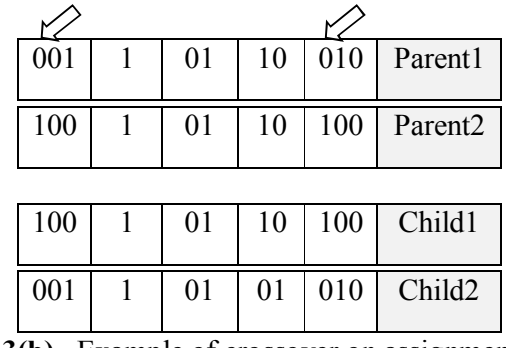

Fig. 3(b). Example of crossover on assignment part

Mutation operator is also performed on two parts of chromosomes. Doing mutation on operation order part, two random numbers are selected: $2 \le r_1 \le r_2 \le (m-1)$ where m is the length of operation order part. The value of substring between two positions is then inverted. For instance, suppose the chromosome (1 2 1 1 2) and *r1* = 2 and *r2* =4 then the new chromosome will be (1 1 1 2 2).

For the machine selection part, a predefined number of operations are selected and for each operation selected, from its available machine list, a machine is randomly selected and assigned to it.

4. Computational results

A metric is defined to test the performance of our algorithm with different parameters. First, for each algorithm, the non-dominated solutions are chosen from all external achievements obtained by the algorithm in all runs and stored in a set H. Then the non-dominated solutions are chosen from the set H. Suppose that n_{tot} is the total number of non-dominated solutions in H, if algorithm Y_i produces n_{Yi} solutions of n_{tot}, the metric q_{Yi} of algorithm Y_i is the ratio of n_{Yi} to n_{tot},

$$p_{Yi} = \frac{n_{Yi}}{n_{tot}}$$

The number of population 50, 60,70,80,90 and 100 are considered and the algorithm is run on 6 problems with various number of jobs and machines. These problems were generated, randomly. Table 1 shows the results. In this Table, the metric were calculated for each problem and each number of populations.

Table 1
P_{Yi} Computational results for 6 problems

		Number of populations					
	Operation × Machines	50	60	70	80	90	100
Ms01	4×5	0.33	0	0.133	0.133	0.133	0.268
Ms02	5×6	0.9	0.1	0	0	0	0
Ms03	7×7	0.367	0.67	0	0.167	0.167	0.1
Ms04	10×10	0.154	0.231	0.154	0.454	0.0776	0.231
Ms05	10×12	0.285	0	0	0.643	0.571	0
Ms06	12×15	0.084	0.416	0.5	0.1	0.134	0.166

As shown in this table, for the problems in small and medium sizes, number of population of 50 is better than others and for problems in big sizes, number of population of 80 is better than others, so for solving the problem in small and medium size we generated 50 populations and for big ones, we generated 80 populations. The proposed algorithm was solved for 10×15 in size problem in Kacem et al. (2002). Final solution is shown in Fig. 4. Three benchmark problems in Kacem et al. (2002) were used for comparing proposed algorithm with such Al+CGA provided by Kacem et al. (2002), PSO+ SA provided by Xia and Wu (2005), PSO+ TS introduced by Gen et al. (2009) and P-DABC is compared by Li et al. (2011). The solutions of our algorithm for 3 benchmark problems consist of Kacem et al. (2002), 3 objective functions (Make span (MS), Total workload (TW), Maximum workload (W) have compared with another algorithm that described in previous paragraph.

Table 2
Comparison results between proposed method and other algorithms

Problem size	Objective functions	AL+CGA			PSO+SA		PSO+TS		P-DABC			Proposed algorithm		
8×8	MS	14	15	16	15	16	14	15	14	15	16	14	15	16
	TW	77	79	75	75	73	77	75	77	75	73	77	72	77
	W	12	13	13	12	13	12	12	12	12	13	12	11	11
10×10	MS	7	7		7		7		8	7	8	7	8	8
	TW	43	45		44		43		41	43	42	42	42	41
	W	5	5		6		6		7	5	5	6	5	7
15×10	MS	11	23	24	12		11		12	11		11	11	
	TW	93	95	91	91		93		91	93		93	91	
	W	11	11	11	11		11		11	11		10	11	

As shown in Table 2, NSGA II algorithm with floating search procedure (our suggested algorithm) has better results than other algorithms. In the case of 8×8 problem size maximum workload objectives function, the results were improved and decreased considerably. The results for 10×10 problem sizes our presented results was better than resemble methods and finally in the 10×15 problem size, not only maximum workload objectives function was improved but also new Pareto solution has superior on the others.

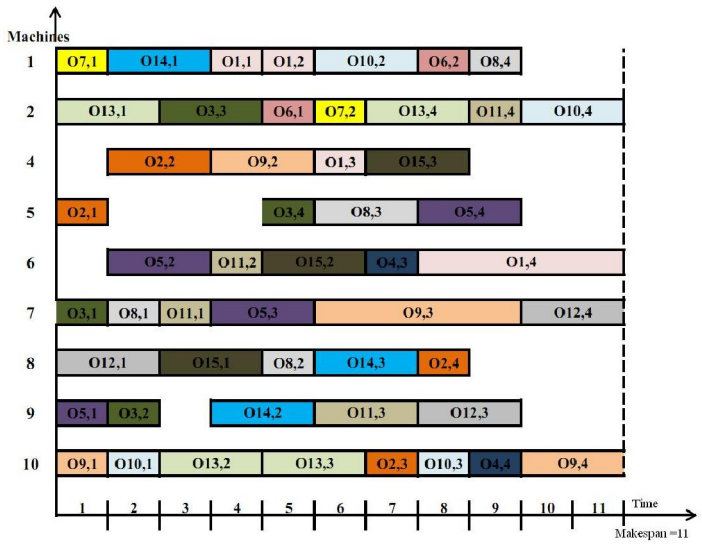

Fig. 4. Final solution for problem 10×15 Kacem et al. (2002)

Metric C is used to compare the approximate Pareto optimal set, respectively, obtained by fore algorithms $C(L,B)$ Measures the fraction of members of B that are dominated by members of L (Zitzler & Thiele1999),

$$C(L,B) = \frac{|\{b \in B : \exists h \in L, h > b\}|}{|B|}$$

If Y1, Y2, Y3,Y4, Y5 are used to denote proposed algorithm, Al+ CGA, PSO+ SA , PSO+ TS and P-DABC, then C_{Y_i,Y_j} indicates the fraction of all non-dominated solutions stored in the archive ofY_j in 20 runs that are dominated by the non-dominated ones obtained by Y_i in all runs. Table 2 shows the computational results. In Table 2, the data in all columns except the first column is related to C_{Y_i,Y_j} and consists of two parts: the first is the value of C_{Y_i,Y_j} and the second the number of non-dominated solutions finally obtained by Y_j after the archive members of Y_i have compared with those of Y_j. As shown in Table 5, for problem 8×8, proposed algorithm produces more non-dominated solutions rather than PSO+SA, PSO+TS and P-DABC, in fig 5 (a) we can see this advantage. Also for problem 10×10 our proposed algorithm has better solutions than AL+CGA and PSO+SA, we can see it in fig 5(b), and for problem 10×15 our proposed algorithm produces more non-dominated solutions than other algorithms and is shown in fig 5(c).

Table 2
Comparing metric C Parameter for five algorithms running on three benchmark problems

		Metric C 1<=>2		Metric C 1<=>3		Metric C 1<=>4		Metric C 1<=>5	
Pt	Damnation	C(Y1,Y2)	C(Y2,Y1)	C(Y1,Y3)	C(Y3,Y1)	C(Y1,Y4)	C(Y4,Y1)	C(Y1,Y5)	C(Y5,Y1)
P1	8*8	0.41	0.62	0.67	0.24	0.75	0.42	0.45	0.3
P2	10*10	0.3	0.12	0.15	0.12	0.23	0.36	0.1	0.2
P3	10*15	0.35	0.15	0.38	0.32	0.62	0.45	0.31	0.3

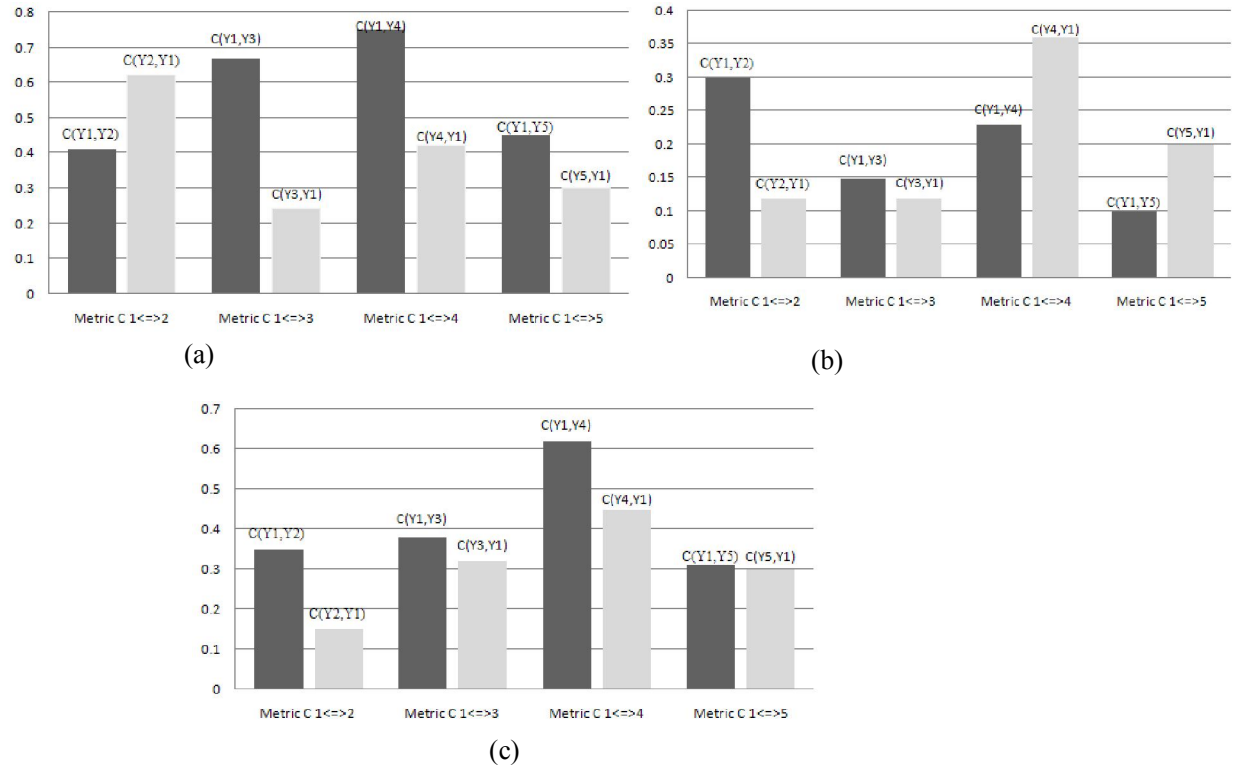

Fig. 5. Final result for problems of Kacem et al. (2002)

5. Conclusion

In this paper, the problem of multi-objective flexible manufacturing job shop was studied. A new approach was presented to improve the search space with floating search to solve the problem. In addition, this approach of wider search in solution space (Likewise the hierarchical approach) decreases the hardness of problem and changes the reach path to final solution. Furthermore, this approach gives useful insight to search upon population and Pareto NSGAII methodologies. Results of experiments suggest an efficient algorithm to reduce variability and to improve pervious methods. Within the formulation, Kacem sample issues were presented in reviewing the efficiency of proposed model and a comparison study has been performed with some recent methods developed by the other researchers. As shown in results, the proposed algorithm has been capable of producing better solutions. Additionally, developing a Pareto set can be followed for the development of the proposed method in future.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, *34*(3), 391-401.

Baykasoğlu, A., & Sönmez, A. İ. (2004). Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *15*(6), 777-785.

Chen, H., Ihlow, J., & Lehmann, C. (1999). A genetic algorithm for flexible job-shop scheduling. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (Vol. 2, pp. 1120-1125). IEEE.

Dauzère-Pérès, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multi-processor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70, 281–306.

Deb, K. (2001), *Multi-Objective Optimization using Evolutionary Algorithms*. Indian Institute of Technology, Kanpur, India.

Fattahi, P., Mehrabad, M. S., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *18*(3), 331-342.

Frutos, M., Olivera, A. C., & Tohmé, F. (2010). A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. *Annals of Operations Research*, *181*(1), 745-765.

Gao, J., Gen, M., Sun, L., & Zhao, X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, *53*(1), 149-162.

Garey, M.R., Johnson, D.S., Sethi, R., (1976). The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research,* 1, 117–129.

Gen, M., Gao, J., & Lin, L. (2009). Multistage-based genetic algorithm for flexible job-shop scheduling problem. In *Intelligent and Evolutionary Systems*(pp. 183-196). Springer Berlin Heidelberg.

Ho, N. B., Tay, J. C., & Lai, E. M. K. (2007). An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, *179*(2), 316-333.

Hurink, E., Jurisch, B., & Thole, M. (1994). Tabu search for the job shop scheduling problem with multi-purpose machines. *Operations Research Spectrum*, 15, 205–215.

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *32*(1), 1-13.

Li, J. Q., Pan, Q. K., & Gao, K. Z. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *55*(9-12), 1159-1169.

Mesghouni, K., Hammadi, S., & Borne, P. (1997, October). Evolution programs for job-shop scheduling. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (Vol. 1, pp. 720-725).

Moradi, E., Fatemi Ghomi, S. M. T., & Zandieh, M. (2011). Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem. *Expert systems with applications*, *38*(6), 7169-7178.

Paulli, J. (1995). A hierarchical approach for the FMS scheduling problem. *European Journal of Operational Research*, 86(1), 32–42.

Saidi-Mehrabad, M., & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithm. *International Journal of Advanced Manufacturing Technology*, 32, 563-570.

Tay, J. C., & Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, *54*(3), 453-473.

Unachak, P., & Adviser-Goodman, E. (2010). An adaptive representation for a genetic algorithm in solving flexible job-shop scheduling and rescheduling problems. Michigan State University in partial fulfillment of the requirements for the degree of DOCTOR OF PHILSOPHY Computer Science.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*(2), 409-425.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, *3*(4), 257-271.