International Journal of Industrial Engineering & Production Research

December 2010, Volume 21, Number 4
pp. 197-209

http://IJIEPR.iust.ac.ir/

ISSN: 2008-4889

Solving Flexible Job Shop Scheduling with Multi Objective Approach

Arash Motaghedi-larijani, Kamyar Sabri-laghaie & Mahdi Heydari*

A. Motaghedi-larijani, *Author is with the Department of Industrial Engineering, Iran University of science & technology , Narmak, Tehran, Iran*
K.. Sabri-laghaie, *Author with the same Department Tehran, Iran*
M. Heydari, *Author is a Faculty at the same Department, Tehran, Iran*

KEYWORDS

Flexible jobshop scheduling,
Multi criteria decision making,
global programming,
Genetic algorithm,
Hill climbing

ABSTRACT

In this paper flexible job-shop scheduling problem (FJSP) is studied in the case of optimizing different contradictory objectives consisting of: (1) minimizing makespan, (2) minimizing total workload, and (3) minimizing workload of the most loaded machine.
As the problem belongs to the class of NP-Hard problems, a new hybrid genetic algorithm is proposed to obtain a large set of Pareto-optimal solutions in a reasonable run time. The algorithm utilizes from a local search heuristic for improving the chance of obtaining more number of global Pareto-optimal solutions. The solution method uses from a perturbed global criterion function for guiding the search direction of the hybrid algorithm. Computational experiences show that the hybrid algorithm has superior performance in contrast to previous studies.

1. Introduction

Finding good solutions to scheduling problems is of great importance to industry, since the production rate and expense of a production plant depend on the schedules used for controlling the work in the plant.
Job-shop scheduling problem (JSP), which is among the hardest combinatorial optimization problems [1], is a branch of production scheduling. It is well known that this problem is NP-hard [2]. The classical JSP schedules a set of jobs on a set of machines with the objective to minimize a certain criterion, subjected to the constraint that each job has a specified processing order through all machines which are fixed and known in advance.
Flexible job-shop problem (FJSP) is an extension of the classical JSP that allows one operation which can be processed on one machine out of a set of alternative machines. So, this kind of scheduling problem reduces machine constraints, and enlarges searching scope of practicable solutions. Flexible job shop scheduling problem have many applications in competitive environment. For example, it is used in flexible manufacturing systems. A flexible manufacturing system consists of several CNC machines. A CNC machine is a multi tasking machine. So flexible job shop scheduling is applicable for flexible manufacturing systems. Bruker and Schlie (1990) were among the first to address this problem [3]. It is closer to the real manufacturing situation. Because of the additional needs to determine the assignment of operations on the machines, FJSP is more complex than JSP, and incorporates all the difficulties and complexities of JSP. The problem of scheduling jobs in FJSP could be decomposed into two sub-problems: a routing sub-problem, which is assigning each operation to a machine out of a set of capable machines and a scheduling sub-problem, which is sequencing the assigned operations on all selected machines in order to obtain a feasible schedule with optimized objectives [4]. For solving the realistic case with more than two jobs, two types of approaches have been used: hierarchical approaches and integrated approaches. In hierarchical approaches assignment of operations to machines and the sequencing of operations on the

* **Corresponding author. M. Heydari**
Email*: a.motaghedi.ie@gmail.com, K_sabri@ind.iust.ac.ir, Mheydari@iust.ac.ir*
Paper first received May. 15. 2010, and in revised form Dec. 05. 2010.

resources or machines are treated separately, i.e., assignment and sequencing are considered independently, where in integrated approaches, assignment and sequencing are not differentiated. Hierarchical approaches are based on the idea of decomposing the original problem in order to reduce its complexity. This type of approach is natural for FJSP since the routing and the scheduling sub-problem can be separated. Brandimarte (1993) was the first to apply the decomposition approach into the FJSP [5]. He solved the routing sub-problem by using some existing dispatching rules and then focused on the scheduling sub-problem, which was solved by using a tabu search heuristic. After that Paulli (1995) applied hierarchical approach [6]. The scheduling problem in a flexible job shop is at least as hard as the job shop problem, which is considered one of the most difficult problems in combinatorial optimization [7]. Zhang and Gen (2005) proposed a method called multistage based genetic algorithm to solve fJSP problem [8]. Saidi Mehrabad and Fattahi (2007) presented a mathematical model and tabu search algorithm to solve the flexible job shop scheduling problem with sequence-dependent setups [9]. They used a hierarchical approach with two heuristics to solve this problem. The first one for assigning each operation to a machine out of a set of two capable machines and the second one for sequencing the assigned operations on all machines in order to obtain a feasible schedule for minimizing the Makespan. Fattahi, Saidi Mehrabad and Jolai (2007) presented a mathematical model and heuristic approaches for flexible job shop scheduling problems [10]. They used mathematical model to achieve optimal solution for small size problems. Since FJSP is NP-hard problem, they developed two heuristic approaches involve of integrated and hierarchical approaches to solve the real size problems and concluded that, the hierarchical algorithms have better performance than integrated algorithms and they used an algorithm which use tabu search and simulated annealing for assignment and sequencing problems consecutively.

Although an optimal solution algorithm for the classical JSP has not been developed yet, there is a tendency to model and solve much more complex version of the FJSP [1]. The research on the multi-objective FJSP is much less than the mono-objective FJSP. Hurink, Jurisch, and Thole (1994) proposed a tabu search heuristic [11].

They considered the reassignment and rescheduling as two different types of moves. Mastrolilli and Gambardella (2000) proposed some neighborhood functions for the FJSP, which could be used in meta-heuristic optimization techniques [12]. The most important issue in employing meta-heuristics for combinatorial optimization problems is to develop an effective ‘‘problem mapping” and ‘‘solution generation” mechanism. If these two mechanisms are devised successfully, it is possible to find good solutions to the given optimization problem in acceptable time. Parsopoulos and Vrahatis (2002) conducted the first research on the particle swarm optimization method in multi-objective optimization problems [13].

Kacem, Hammadi, and Borne (2002) proposed a genetic algorithm controlled by the assignment model which was generated by the approach of localization (AL) to mono-objective and multi-objective FJSP [14,15]. They used the integrated approach considering assignment and scheduling at the same time. Rigao (2004) developed two heuristics based on tabu search: a hierarchical procedure and a multiple start procedure [16]. Recently, Xia and Wu (2005) proposed a hybrid algorithm using particle swarm optimization (PSO) assignment and simulated annealing (SA) scheduling to optimize multi-objective FJSP, respectively [4]. Low , Yip and WU (2006) made use of one of the multiple objective decision-making methods, a global criterion approach, to develop a multi-objective model for solving FMS scheduling problems with consideration of three objectives [17].

Zhang, Zheng, and Wu (2007) proposed a hybrid of ant colony and particle swarm optimization algorithms to solve the multi-objective flexible job-shop scheduling problem based on the analysis of objectives and their relationship [18]. Gao, Gen, Sun and Zhao (2007) developed a new genetic algorithm hybridized with an innovative local search procedure (bottleneck shifting) for the problem [19]. Xing, Chen and Yang(2009) constructed a knowledge-based heuristic algorithm which combined heuristic algorithm with empirical knowledge for the multi-objective flexible job shop scheduling problem [20]. Also Xing, Chen and Yang (2009) presented a simulation model to solve the multi-objective flexible job shop scheduling problem [21].

To our knowledge, research on multi-objective FJSP is rather limited, and most traditional optimal approaches used only one optimization algorithm for solving multi-objective FJSP, thereupon in this paper, an effective hybrid genetic algorithm is proposed to solve the multi-objective flexible jobshop scheduling problems and accordingly comparisons with other similar works is added.

In this research a memetic algorithm is applied to a flexible job shop scheduling problem in which three contradictory objectives of makespan, the total workload of machines and maximum workload of the machines has been considered. Unlike other similar works that have integrated the contradictory objectives by weighting method, a compromise programming approach is used to deal with these three objectives. The effect of applying compromise programming method is compared with weighting method.

The remainder of this paper is organized as follows: The notation and problem description are introduced in section 2. Section 3 presents method of the global criterion. Section 4 describes genetic algorithm for global search and hill climbing for local search. In

section 5, computational experiments performed with proposed approach are reported followed by the comparison with other heuristic methods. Finally some concluding remarks are made in section 6.

2. Problem Formulation

In the flexible job shop scheduling, we have two challenges. The first one is to assign each operation to a machine, and the second one is to sequence the operations on each machine in order to minimize makespan. The flexible job shop scheduling problem can be defined by:

- There are n jobs, indexed by i, and these jobs are independent one of each other.
- Each job i has an operating sequence, denoted by J_i (precedence constraints). J_i also denotes the i-th job if no confusion.
- Each operating sequence is an ordered set of operations $O_{i,j}$ for $j = 1,\ldots, l_i$.
- There are m machines, indexed by k (the k-th machine is denoted by M_k).
- For each operation $O_{i,j}$, there is a set of machines capable of performing it. The set is denoted by $M_{i,j}$, $M_{i,j} \in \{1,\ldots,m\}$ (routing constraints), if there is.
- $M_{i,j} \in M$ for at least one operation, it is partial flexibility FJSP (P-FJSP); while there is $M_{i,j} = M$ for each operation, it is total flexibility FJSP (T-FJSP) [14,15].
- The processing time of an operation $O_{i,j}$ on machine k is predefined and denoted by $t_{i,j,k}$.

Tab. 1. Problem 4×5 with 12 operations (total flexibility)

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Tab. 2. Problem 8×8 with 27 operations (partial flexibility)

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
J_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
J_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
J_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

In this paper, the following criteria are to be minimized:
(1) the maximal completion time of machines, i.e., the makespan (f_1);
(2) the maximal machine workload, i.e., the maximum working time spent on each machine (f_2);
(3) the total workload of machines, which represents the total working time over all machines (f_3).
In scheduling problems makespan have a key role and it is used in objective function. If makespan reduce, the order can be delivered to customer earlier. Second criterion balances working time spent on each machine. Finally third criterion leads to reduce the total working time over all machines. It is clear that all criteria conflict together. For example, if second criterion reduces third criterion maybe increases or minimizing makespan lead to increasing second criterion. During the process of solving this problem, the following assumptions are made:
(1) Each operation cannot be interrupted during its performance (non-preemptive condition);

(2) Each machine can perform at most one operation at any time (resource constraint);
(3) Each machine becomes available to other operations once the operations which are currently assigned to be completed.
(4) All machines are available at t = 0.
(5) All jobs can be started at t = 0.
(6) The precedence constraints of the operations in a job can be defined for any pair of operations;
(7) Setting up time of machines and move time between operations are negligible;
(8) Machines are independent from each other;
(9) There are no precedence constraints among operations of different jobs.
(10) Neither release times nor due dates are specified.

Tab. 3. Problem 10×10 with 30 operations (total flexibility)

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3

3. Method of the Global Criterion

The method of the global criterion is also sometimes called compromise programming In this method, the distance between some reference point and the feasible objective region is minimized. The analyst has to select the reference point and the metric for measuring the distances. All the objective functions are thought to be equally important.

3-1. Different Metrics

Here we examine the method where the ideal objective vector is used as a reference point and L_p-metrics are used for measuring. In this case, the L_p-problem to be solved is:

$$\text{Minimize} \quad \left(\sum_{i=1}^{k} \left| f_i(x) - z_i^* \right|^p \right)^{1/p} \qquad (1)$$

Subject to $x \in S$

From the definition of the ideal objective vector z_i^* we know that $f_i(x) \geq z_i^*$ for all $i = 1, \cdots, k$ and all $x \in S$. This is why no absolute values are needed if we know the global ideal objective vector. If the global objective vector is not known, the method does not necessarily work as it should. If the ideal objective vector is replaced by some other vector, it must be selected carefully. Pessimistic reference points must be avoided since the method cannot find solutions better than reference point. We should know that, the solution obtained by the model depends greatly on the value chosen for p. Widely used choices are. $p = 1, 2 \; or \; \infty$.

Instead of the terms $\left| f_i(x) - z_i^* \right|$, denominators may be added to problem (1.1) to normalize the components, that is, to use the terms $\left| f_i(x) - z_i^* \right| / \left| z_i^* \right|$ instead. Some other denominators, like $\left| z_i^{nad} - z_i^* \right|$, can also be used. The reason for employing denominators is that sometimes it is worthwhile to use relative distances in the calculations. For example, using the components of z_i^* forms the contour of the metric to reflect better the location of the ideal objective vector. Naturally, the denominators z_i^* cannot be used if some of them equals zero [22].

Briefly, the method of the global criterion is a simple method to use if the aim is simply to obtain a solution where no special hopes are set. This is a method that combines all objective functions to form a single function. Advantages of the method are: (1) it focuses on a single objective with limits on others, (2) it always provides a weakly Pareto optimal point, assuming the formulation gives a solution, (3) it is not necessary to normalize the objective functions, (4) it gives Pareto optimal solution if one exists and is unique. The only disadvantage is that the optimization problem may be infeasible if the bounds on the objective functions are not appropriate.

4. Hybrid Algorithm With Hill Climbing Approach

4.1 Genetic Algorithms

Genetic algorithm (GA) is a stochastic search method for optimization problems based on the mechanics of natural selection and natural genetics (i.e., survival of the fittest). GA has demonstrated considerable success in providing good solutions to many complex optimization problems and received more and more attentions during the past three decades. When the objective functions to be optimized in the optimization problems are multimodal or the search spaces are particularly irregular, algorithms need to be highly robust in order to avoid getting stuck at a local optimal solution. The advantage of GA is just able to obtain the global optimal solution fairly. In addition, GA does not require the specific mathematical analysis of optimization problems, which makes GA easily coded by users who are not necessarily good at mathematics and algorithms. One of the important technical terms in

GA is chromosome, which is usually a string of symbols or numbers. A chromosome is a coding of a solution of an optimization problem, not necessarily the solution itself. GA starts with an initial set of randomly generated chromosomes called a population. The number of individuals in the population is a predetermined integer and is called population size. All chromosomes are evaluated by the so-called evaluation function, which is some measure of fitness. A new population will be formed by a selection process using some sampling mechanism based on the fitness values. The cycle from one population to the next one is called a generation. In each new generation, all chromosomes will be updated by the crossover and mutation operations. The revised chromosomes are also called offspring. The selection process selects chromosomes to form a new population and the genetic system enters a new generation. After performing the genetic system a given number of cycles, we decode the best chromosome into a solution which is regarded as the optimal solution of the optimization problem [23]. The aim of this section is to introduce an effective GA for solving complex optimization problems. The overall structure of our GA can be described as follows:

4.1.1. Initial Population:
The initial chromosomes are obtained by a random procedure.

4.1.2. Fitness Function:
Chromosomes in the population will be evaluated by the objective function which is a combination of f1, f_2 and f_3 according to the description in Section "problem formulation". In this paper the global criterion method is used to transform the different objectives into one objective. Accordingly, our combined objective function can be expressed as,

$$Minimize \quad F=\frac{\left|f_1(x)-z_1^*\right|}{\left|z_1^*\right|}+\frac{\left|f_2(x)-z_2^*\right|}{\left|z_2^*\right|}+\frac{\left|f_3(x)-z_3^*\right|}{\left|z_3^*\right|}$$

Where $z^*=(z_1^*,z_2^*,z_3^*)$, as explained in section "Method of global criterion", is the ideal vector. In order to have a good estimation of ideal vector, in this paper, we have run a genetic algorithm for each individual objective function to obtain z^* vector.

Tab. 4. Problem 15×10 with 56 operations (total flexibility)

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
J_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
J_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	O_{34}	7	3	12	1	6	5	8	3	5	2
J_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
J_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
J_6	$O_{6.1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6.2}$	1	3	6	5	4	7	5	4	6	5
J_7	$O_{7,1}$	1	4	2	5	3	6	9	8	5	4
	$O_{7,2}$	2	1	4	5	2	3	5	4	2	5

Continuance Tab. 4. Problem 15×10 with 56 operations (total flexibility)

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2
J_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	10	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
	$O_{9,4}$	9	8	7	4	5	8	7	4	56	2
J_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
J_{11}	$O_{11,1}$	1	2	3	6	5	2	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4
J_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
J_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
J_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
J_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5
	$O_{15,3}$	4	5	2	3	5	2	8	4	7	5
	$O_{15,4}$	6	2	11	14	2	3	6	5	4	8

Representation of A-string

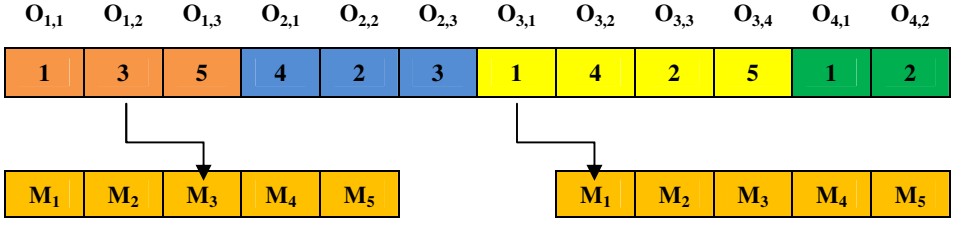

Fig. 1. Representation of A-string

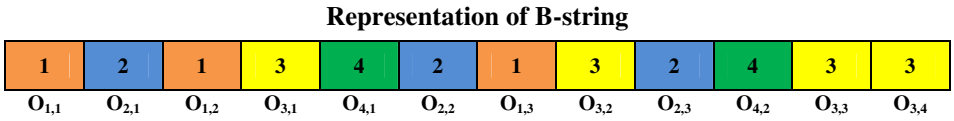

Fig. 2. Representation of B-string

In a generation, after the evaluation of each chromosome, the best fitness will be the lowest one and should be transferred to the next generation during the search process.

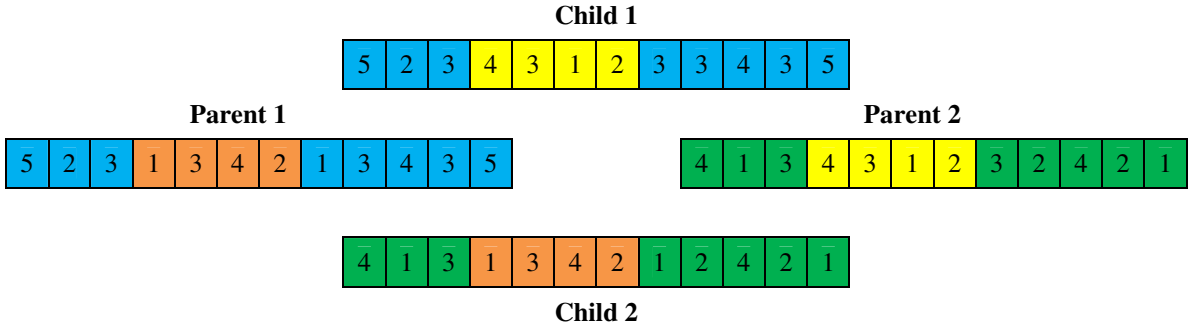

Fig. 3. Cross over of A-string

4.1.3. Encoding Scheme:

In order to apply GA successfully to a multi-objective FJSP problem, appropriate representation of chromosomes is of a great importance. In this matter, Zhang G. et al.(2008) divided the chromosomes into two parts, A-string and B-string. In this improved representation, A-string represents machines assigned to operations and B-string defines the sequence of operations. On the other hand in this representation the feasibility is considered automatically [24]. So there is no need of any repair mechanism to maintain feasibility. For instance, consider the problem in the table 1. The problem is to execute 4 jobs on 5 machines. An instance of a possible encoding of A and B-string can be seen in Fig. 1 and Fig. 2 respectively.

The elements of A-string represent the machines assigned to operations. The first element is representative of operation $O_{1,1}$, the second is representative of operation $O_{1,2}$ and in like manner the last element is representative of operation $O_{4,2}$. The number located in each element is the number of machine assigned to each operation.

B-string is as long as A-string. Numbers in B-string show a sequence of job numbers in which job number i occurs n_i times. The schedule is always feasible if each operation is replaced by the corresponding job index. As illustrated in Fig 2, one possible B-string may be 1-2-1-3-4-2-1-3-2-4-3-3. The list of ordered operations related to this B-string could be translated as: $O_{1,1}$- $O_{2,1}$- $O_{1,2}$- $O_{3,1}$- $O_{4,1}$- $O_{2,2}$- $O_{1,3}$- $O_{3,2}$- $O_{2,3}$- $O_{4,2}$- $O_{3,3}$- $O_{3,4}$. When an individual is decoded, at first the operations are assigned to machines according to A-string.

4.1.4. Crossover Operations:

Crossover is the process of taking two parent solutions and producing from them a child. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Therefore, Crossover is a recombination operator that proceeds in three steps:

i. The reproduction operator selects at random a pair of two individual strings for the mating.

ii. A cross site is selected at random along the string length.

iii. Finally, the position values are swapped between the two strings following the cross site [23]. In this hybrid algorithm the crossover operations of both A-string and B-string perform separately, and their performance is different. According to the method that [24] used, the crossover operation for A and B-string is as follows. For A-string two positions are randomly chosen from the parents and the whole substrings between these two positions are swapped and then two new strings are created.

The procedure could be illustrated as Fig 3. In order to perform crossover operation on B-string, we divide all jobs into two groups by a random procedure, J1 and J2. At the next stage, we retain the jobs from parent 1 that belong to J1 and we do so for parent 2 and J2. Then we remove the other elements of the parents. The remaining elements of parent 1 and parent 2 are respectively transferred to child 1 and child 2 with the same positions. The empty positions of child 1 and child 2 will be orderly filled with the elements of parent 2 and parent 1 that belong to their previous sequence. This procedure could be illustrated as Fig. 5.

4.1.5. Mutation:

Another important genetic operator is mutation, Mutation prevents the algorithm to be trapped in a local minimum. Mutation plays the role of recovering the lost genetic materials as well as for randomly disturbing genetic information. It is an insurance policy against the irreversible loss of genetic material Mutation is viewed as a background operator to maintain genetic diversity in the population.

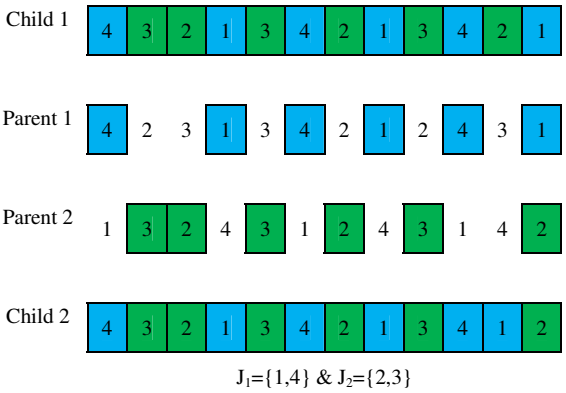

Fig. 4. Cross over of B-string

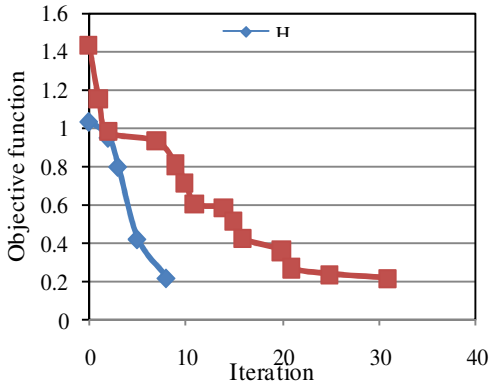

Fig. 5. Comparison of GA and HGA performance on problem 4×5

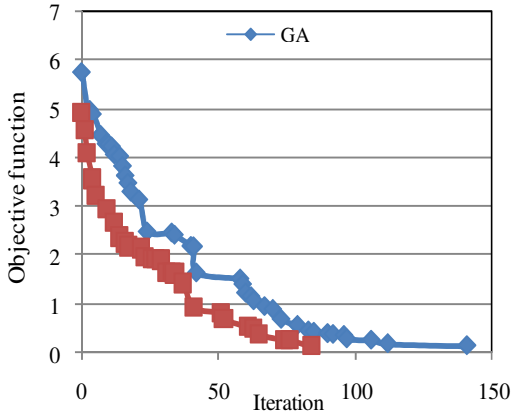

Fig. 6. Comparison of GA and HGA performance on problem 8×8

It introduces new genetic structures in the population by randomly modifying some of its building blocks. Mutation helps escape from local minima's trap and maintains diversity in the population. The important parameter in the mutation technique is the mutation probability.

The mutation probability decides how often parts of chromosome will be mutated. Mutation is applied with small probability [23]. Here is presented just one mutation operator which is performed by the B-string of each chromosome.

Zhang G. et al(2008) used the following procedure; by generating two random numbers, two positions of B-string will be selected, then the substring between this two positions will be inverted to get a new B-string. The new B-string will be replaced with the old one [24].

4.2. Hill Climbing Method

In computer science, hill climbing is a mathematical optimization technique which belongs to the family of local search. It is relatively simple to implement, making it a popular first choice. Although more advanced algorithms may give better results, in some situations hill climbing works just as well.

Hill climbing can be used to solve problems that have many solutions, some of which are better than others. It starts with a random (potentially poor) solution, and iteratively makes small changes to the solution, each time improving it a little.

When the algorithm cannot see any improvement anymore, it terminates. Ideally, at that point the current solution is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution.

In simple hill climbing, the first closer node is chosen, whereas in steepest ascent hill climbing all successors are compared and the closest to the solution is chosen. Both forms fail if there is no closer node, which may happen if there are local maxima in the search space which are not solutions. Steepest ascent hill climbing is similar to best-first search, which tries all possible extensions of the current path instead of only one [25].

In this paper, hill climbing approach is used to improve derived solutions of the GA.

In fact, hill climbing method is applied to change the order of assigned jobs. With regard to neighborhood function hill climbing approach looks for the best solution.

In this algorithm, neighborhood function is just defined to change the sequence of assigned jobs to each machine. Hence, algorithm is just able to improve C_{max} and doesn't change Max WL and $\sum$WL.

4.3. Hybrid Algorithm:

Now the framework of hybrid algorithm is explained. As illustrated in Fig. 8 it consists of five major steps.

Inputs {GA parameters}

Step 1. Generate initial population randomly according to the encoding scheme.

Step 2. Compute C_{max} , Max WL, $\sum$WL and then evaluate each chromosome by fitness function.

Step 3. Perform crossover and mutation operations on the population at hand and then generate new population.

Step 4. Perform hill climbing procedure on each chromosome.

(*a*) Search the neighborhood subset to find the best possible solution.

(*b*) If the solution is of better C_{max} then

repeat (*a*),

else

go to step 5.

Step 5. Extract the best found fitness function.

Step 6. If the termination criterion is not met go to step 4; otherwise, go to step 3. The termination criterion is a predefined number of generations.

Fig. 7. The structure of hybrid algorithm

5. Experimental Results

To illustrate the effectiveness and performance of the hybrid algorithm proposed in this paper, the algorithm procedure was implemented in Java eclipse on a vostro 1500 with 2.2 GHz CPU and 2 GB Ram. Four representative instances based on practical data have been selected. Tables 1 to 4 represent the mentioned instances. Each instance can be characterized by the following parameters: number of jobs (n), number of machines (m), and each operation $O_{i,j}$ of job i. Four problem instances (problem 4 × 5, problem 8 × 8, problem 10 × 10, and problem 15 × 10) are all taken from [14,15].

In tables 5 ,6 ,7 and 8 the column labeled additive objective refers to the objective function of weighted method that objective functions are weighted equally.

The Hybrid Algorithm is compared with following published algorithms in the comparative studies. The comparative results are shown in Tables 5, 6, 7, 8. For these tables, rows have been labeled as:

(1) 'Star' refers to the best solution found for each single objective function .

(2) A1: 'Temporal Decomposition' refers to [14];

(3) A2: 'Classic' GA refers to [14];

(4) A3: Approach by Localization refers to [14];

(5) A4: 'AL+CGA' refers to [14];

(6) A6: 'PSO+SA' refers to [4];

(7) 'moGA' refers to [8];

(8) 'hGA' refers to [19];

(9) 'PSO+TS' refers to [24];

(10) 'simulation modeling' refers to [21];

(11) 'GA+HC' refers to our Hybrid algorithm

In order to attain the star objectives that that are the best solutions found for each single objective function, every instance is run for 100 times. Then each instance according to its scale is run between 10-50 times and all among solutions with the best objective function the most repeated solution is returned.

5.1. Problem 4 × 5

At first, we use a small scale instance to display the optimizing ability and evaluate the effectiveness of this hybrid algorithm. This instance is taken from [15]. We have applied it to the instance in Table 1. The GA parameters are as follow: population size=200, iteration number=100, mutation rate=0.1. The best values by the hybrid algorithm are shown as follows:

Solution1: Makespan = 13, Max(Wtw) = 7, Wtw = 33
Solution2: Makespan = 12, Max(Wtw) = 8, Wtw = 32

Where Makespan is maximum complete time, Max(Wtw) is used to represent the critical machine workload, and Wtw represents the total workload of all machines. From Table 5 we could obviously see that the proposed hybrid algorithm has reduced two objective under the same value of Max(Wtw): the makespan (11 instead of 13) and the Wtw (32 instead of 34). By comparing between the data in these two columns, it is clearly shown that the proposed novel hybrid algorithm is effective.

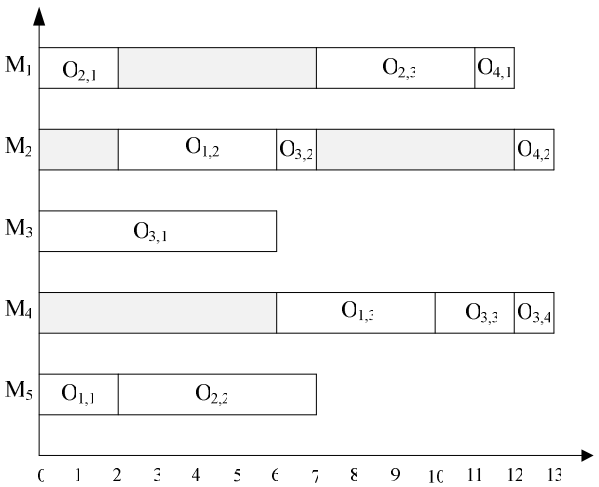

Fig. 8. Optimization solution of problem 4×5

5.2. Problem 8×8

Table 2 displays the instance of the middle scale problem 8×8. The GA parameters are as follow: population size=100, iteration number=500, mutation

rate=0.1. The computational results by the proposed hybrid algorithm are shown in the following:

Solution 1: Makespan =14, Max(Wtw) = 11, Wtw = 78
Solution 2: Makespan =14, Max(Wtw) = 12, Wtw = 77

Tab. 5. Comparison of results on problem 4 * 5

Objectives	Makespan	Max(Wtw)	Wtw	Additive Objective	Global Objective
Stars	11	7	32	-	-
AL + GA	16	10	34	20	0.9456
PSO + S	11	10	32	17.67	0.4286
GA+	13	7	33	17.67	0.2131
HC	12	8	32	17.33	0.2338

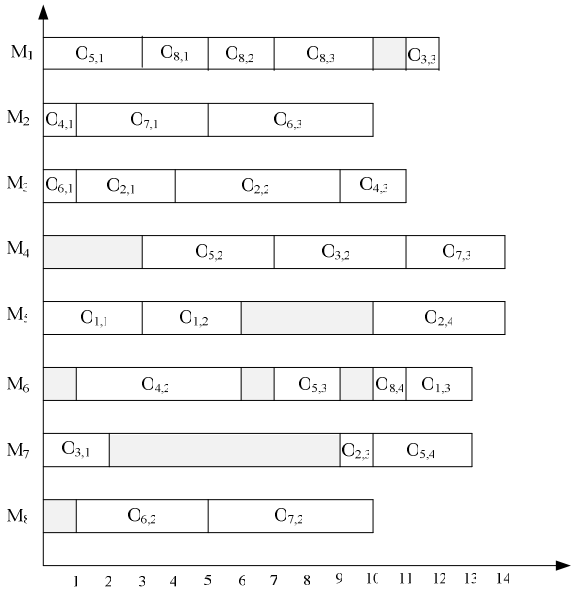

Fig. 9. Optimization solution of problem 8×8

Tab. 6. Comparison of results on problem 8 * 8

objectives	Makespan	Max(Wtw)	Wtw	Additive Objective	Global criterion
star	14	11	73	32.67	-
Temporal decomposition	19	19	91	43	1.331
Classic GA	16	11	77	34.67	0.1976
Approach by localization	16	13	75	34.67	0.3521
AL + CGA	15	13	79	35.67	0.3354
	16	13	75	34.67	0.3521
PSO + SA	14	12	77	34.33	0.1457
	15	12	75	34	0.1897
moGA	15	14	73	34	0.2857
PSO + TS	14	12	77	34.33	0.1457
	15	12	75	34	0.1897
hGA	15	12	75	34	0.1897
simulation modeling	14	12	77	34.33	0.1457
	15	12	76	34.33	0.2034
GA+HC	14	11	78	34.33	0.0685
	14	12	75	33.67	0.1183

The solution 1 has been graphically represented in Fig. 9. The comparison of our hybrid algorithm with other algorithm is shown in Table 6

5.3. Problem 10×10

A middle scale instance which is taken from [15] is shown in Table 3. And it was taken to evaluate the efficiency of proposed hybrid algorithm. The GA parameters are as follow: population size=500, iteration number=500, mutation rate=0.1 The computational results obtained by the proposed hybrid algorithm are given in the following:
Solution1: Makespan = 7, Max(Wtw) = 5, Wtw = 43
Solution2: Makespan = 8, Max(Wtw) = 5, Wtw = 42
The schedule's Gantt chart representation corresponding to the solution is shown in Fig. 10. The comparison of the proposed hybrid algorithm with other algorithms is shown in Table 7.

Tab. 7. Comparison of results on problem 10 * 10

Objectives	Makespan	Max(Wtw)	Wtw	Additive Objective	Global criterion
Stars	7	5	41	-	-
Temporal decomposition	16	16	59	30.33	3.9247
Classic GA	7	7	53	22.33	0.6927
Approach by localization	8	6	46	20	0.4648
AL + CGA	7	5	45	19	0.0976
PSO + SA	7	6	44	19	0.2732
moGA	7	5	43	18.33	0.0488
hGA	7	5	43	18.33	0.0488
PSO + TS	7	6	43	18.67	0.2488
	8	6	42	18.67	0.3672
Simulation modeling	8	5	42	18.33	0.1672
	7	6	42	18.33	0.2244
GA+HC	7	5	43	18.33	0.0488
	8	5	42	18.33	0.1672

Tab. 8. Comparison of results on problem 15 * 10

Objectives	Makespan	Max(Wtw)	Wtw	Additive Objective	Global criterion
Star	11	10	91	-	-
AL + CGA	24	11	91	42	1.2818
	23	11	95	43	1.2349
PSO + SA	12	11	91	38	0.1909
hGA	11	11	91	37.67	0.1
PSO + TS	11	11	93	38.33	0.122
	12	11	92	38.33	0.2019
Simulation modeling	11	11	92	38	0.111
	11	10	93	38	0.022
GA+HC	11	11	91	37.67	0.1
	11	11	91	37.67	0.1
	12	10	93	38.33	0.1129

5.4. Problem 15×10

To evaluate the efficiency and validity of our algorithm on larger scale problems, we have chosen the instance shown in Table 4 which is taken from [15]. The GA parameters are as follow: population size=600, iteration number=500, mutation rate=0.1. The best computational results by this hybrid algorithm are shown in the following:

Solution1: Makespan = 11, Max(Wtw) = 11, Wtw = 91
Solution2: Makespan = 12, Max(Wtw) = 10, Wtw = 93

The computational result's schedule is represented by Gantt chart in Fig. 11.The comparison of proposed hybrid algorithm with other algorithm is shown in Table 8.

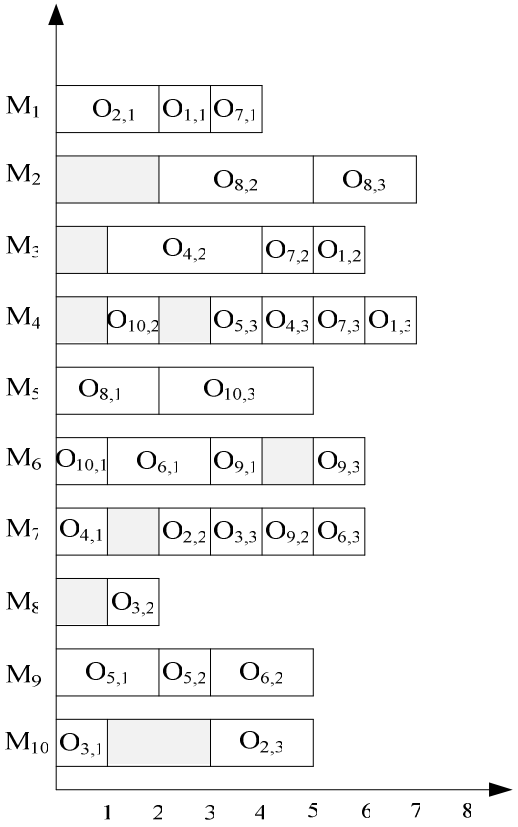

Fig. 10. Optimization solution of problem 10×10

6. Future Research and Conclusion

In this paper, an effective hybrid genetic algorithm is proposed to solve the multi-objective flexible jobshop scheduling problems. In this hybrid algorithm, genetic algorithm perform global search and hill climbing is used to local search. When the initial answer was obtained by the GA, local search starts with Hill climbing. the assignment is considered as a neighborhood structure for the problem. The performance of the presented algorithm is evaluated in comparison with the results obtained from other authors' algorithms for four representative instances. The obtained results show the effectiveness of the proposed algorithm. Also Effect of neighborhood searching by tables 5, 6, 7 and 8 has been shown.

In this paper, a Pareto approach is proposed to solve the multi objective flexible job shop scheduling problems. The objectives considered are to minimize the maximal completion time of machines, the maximal machine workload and the total workload of machines.

This approach gets idea from compromise programming. First the problem have been solved by considering only one objective function and obtained solution is named star solution. Then global objective function is made.

The contribution of this paper can be summarized as follows:

- It presented a hybrid algorithm to solve the flexible job shop scheduling problems with multi-objectives of minimizing makespan, the total workload of machines and maximum workload of the machines
- In this paper multi objective FJSP is solved by method that get idea from compromise programming. This method is more reasonable with comparison of weighting method.
- The effect of local search has been compared with genetic algorithm only. Figures 6,7, 12 and 13 have illustrated that hybrid algorithm can be reached to acceptable solution in less iteration.

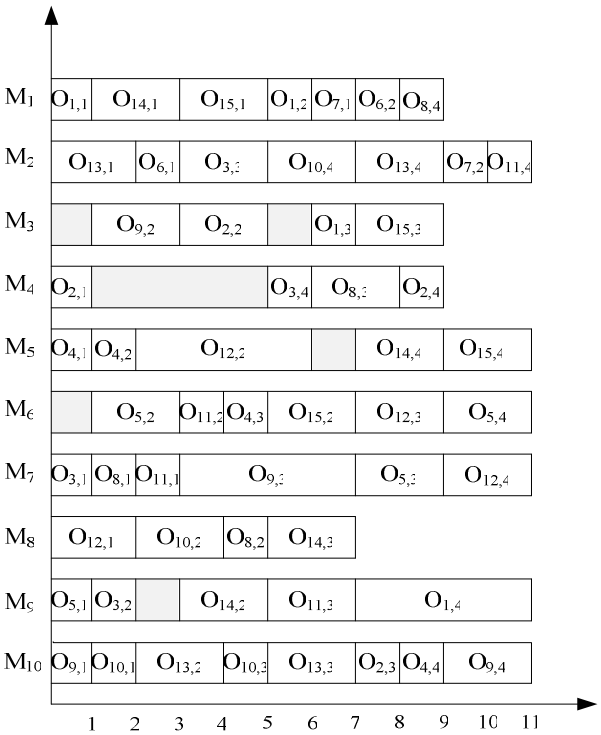

Fig. 11. Optimization solution of problem 15×10

The future researches of this paper are as follows:

- Improving the local search by defining new neighborhood structures.
- Eliminating one of the assumption in this paper. For example, eliminating non preemptive condition. It can be enlarge solution space.
- Proposing an approach that can avoid local optimum solutions.

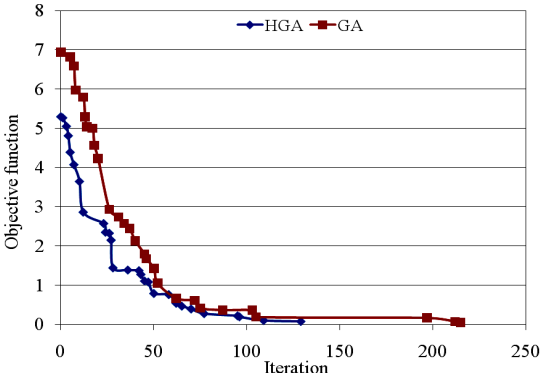

Fig. 12. Comparison of GA and HGA performance on problem 10×10

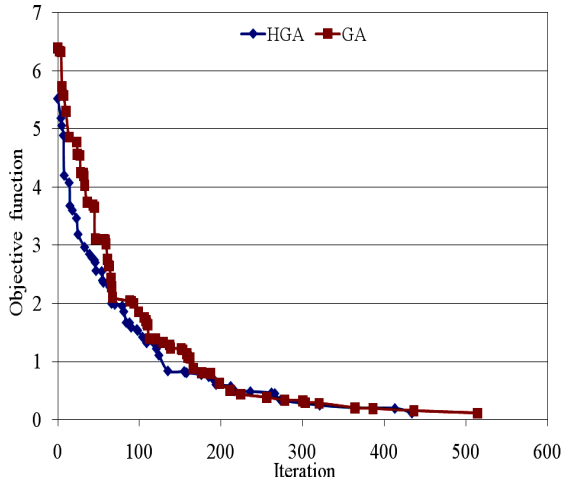

Fig. 13. Comparison of GA and HGA performance on problem 15×10

References

[1] Baykasoglu, A., Ozbakir, L., Sonmez, A.I., *"Using Multiple Objective Tabu Search and Grammars to Model and Solve Multi-Objective Flexible Job-Shop Scheduling Problems."* Journal of Intelligent Manufacturing, Vol. 15(6), 2004, pp. 777–785.

[2] Garey, M.R., Johnson, D.S., Sethi, R., *"The Xomplexity of Flow Shop and Jobshop Scheduling,"* Mathematics of Operations Research, Vol. 1, 1976, pp. 117–129.

[3] Bruker, P., Schlie, R., "Job *Shop Scheduling with Multi-Purpose Machine",* Computing, Vol. 45, 1990, pp. 369–375.

[4] Xia, W.J., Wu, Z.M., *"An Effective Hybrid Optimization Approach for Multiobjective Flexible Job-Shop Scheduling problems,"* Computers and Industrial Engineering, Vol. 48(2), 2005, pp. 409–425.

[5] Brandimarte, P., *"Routing and Scheduling in a Flexible Job Shop by Taboo Search,".* Annals of Operations Research, Vol. 41(3), 1993, pp. 157–183.

[6] Paulli, J., *"A Hierarchical Approach for the FMS Scheduling Problem,"* European Journal of Operational Research Vol. 86, 1995, pp. 32-42.

[7] Lawler, E.l., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., "*Sequencing and Scheduling: Algorithms and Complexity,*" In S. C. Graves et al. (Eds.), Logistics of production and inventory Amsterdam: North Holland, 1993, pp. 445–522.

[8] Zhang, H., Gen, M., *"Multistage-Based Genetic Algorithm for Flexible Job-Shop Scheduling Problem,"* Journal of Complexity International, Vol. 11, 2005, pp. 223–232.

[9] Saidi Mehrabad, M., Fattahi, P., *"Flexible Job Shop Scheduling with Tabu Search Algorithm."* International Journal of Advanced Manufacturing Technology, Vol. 32, 2007, pp. 563–570.

[10] Fattahi, P., Saidi Mehrabad, M., Jolai, F., *"Mathematical Modeling and Heuristic Approaches to Flexible job Shop Scheduling Problem,"* Journal of Intelligent Manufacturing, Vol. 18, 2007, pp. 331–342.

[11] Hurink, E., Jurisch, B., Thole, M., *" Tabu Search for the Job Shop Scheduling Problem with Multi-Purpose Machine,"* Operations Research Spektrum, Vol. 15 , 1994, pp. 205–215.

[12] Mastrolilli, M., Gambardella, L.M., *"Effective Neighborhood Functions for the Flexible Job Shop Problem,"* Journal of Scheduling, Vol. 3(1), 2000, pp. 3–20.

[13] Parsopoulos, K.E., Vrahatis, M.N., *"Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization.* Natural Computing," Vol. 1(2-3), 2002, pp. 235–306.

[14] Kacem, I., Hammadi, S., Borne, P., *"Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problem,"* IEEE transactions on systems, man, and cybernetics, Part C, Vol. 32(1), 2002, pp. 1–13.

[15] Kacem, I., Hammadi, S., Borne, P., *"Pareto-Optimality Approach for Filexible Job-Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic,"* Mathematics and Computers in Simulation, Vol. 60, 2002, pp. 245–276.

[16] Rigao, C., *"Tardiness Minimization in a Flexible Job Shop: A Tabu Search Approach,"* Journal of Intelligent Manufacturing, Vol. 15(1), 2004, pp. 103–115.

[17] Low C., Yip Y., Wu T.H., *"Modeling and Heuristics of FMS Scheduling with Multiple Objectives,"* Computers and Operations Research, Vol. 33(3), 2006, pp. 674-694.

[18] Zhang, W.C., Zheng, P.E., Wu, X.D, *"Solution to Flexible Job Shop Scheduling Problems with Capacitated Constraints Based on ant Colony & Genetic Algorithms,"* Computer Integrated Manufacturing Systems, Vol. 13(2), 2007, pp. 333–337.

[19] Gao, J, Gen, M., Sun, L., Zhao, X., *"A Hybrid of Genetic Algorithm and Bottleneck Shifting for Multiobjective Flexible Job Shop Scheduling Problems,"* Computers & Industrial Engineering, Vol. 53, 2007, pp. 149–162.

[20] Xing, L.N., Chen, Y.W., Yang, K.W., *"An Efficient Search Method for Multi-Objective Flexible Job Shop Scheduling Problems,"* Journal of Intelligent Manufacturing, Vol. 20, 2009, pp. 283–293.

[21] Xing, L.N., Chen, Y.W., Yang, K.W., *"Multi-Objective Flexible Job Shop Schedule: Design and Evaluation by Simulation Modeling,"* Applied Soft Computing, Vol. 9, 2009, pp. 362–376.

[22] S.Arora, J., Introduction to optimum design (2nd edition), Elsevier Press, 2004, pp. 556-559.

[23] Baoding, Liu., Theory and Practice of Uncertain Programming (2nd edition), Genetic Algorithms. Springer-Verlag Berlin Heidelberg, 2009, pp. 9-14.

[24] Zhang, G., Shao, X., Li, P., Gao, L, "*An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem,"* Computers & Industrial Engineering, Vol. 56(4), 2008, pp. 1309-1318.

[25] Meittinen, K., Nonlinear Multiobjective Optimization. No-Preference Methods. Kluwer Academic Publishers. 1999, pp. 67-71.