J. Basic. Appl. Sci. Res., 3(9)713-720, 2013
© 2013, TextRoad Publication
ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Solving the Multi Objective Flexible Job Shop Problem Using Combinational Meta Heuristic Algorithm Based on Genetic Algorithm and Tabu-Search

[1.]Hamid Reza Mollaei[*], [2.]Samira Zeynali, [3.]Nasser Shahsavari Pour

[1*]Department of Industrial Management, Science and Research Branch, Islamic Azad University, Kerman, Iran.
[2.]Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Kerman, Iran.
[3.]Department of Industrial Management, Vali-e-Asr University, Rafsanjan, Iran.

ABSTRACT

Flexible Job Shop Scheduling Problem (FJSSP) has an especial place in industry environments; due to this issue and also due to its mathematical characteristics large number of managers and researchers are considered this problem. Flexible Job Shop Scheduling Problem is more complexthan JSSP and classified as Np- Hard problems. So in this paper a combinational optimization Meta heuristic based on Genetic algorithm is proposed for solving this problem and Tabu search method as a local search algorithm is used to increasing the quality of solutions. In current paper the FJSSP is studied in Multi objective mode and during the solving the problem three objective functions are considered as Makespan, Total workload of machines and Maximum workload of machines and all of them should be minimized.
This problem is coded by VBA Software and finally the solutions of proposed algorithm are compared with other papers and the efficiency of solution will be examined.
KEY WORDS: Flexible Job Shop Scheduling problem, combinational optimization, Genetic algorithm, Tabu search, Multi objective

1. INTRODUCTION

Flexible job shop scheduling problem (FJSSP) is one of the most important fields in both of production management and combinatorial optimization. It has more usage than job shop systems problem and through that we can use for increasing production, solving bottleneck problem, or use as a competitive issue in economic environment[1]. In the static JSSP, a finite number of jobs are to be processed by a finite number of machines. Each job consists of a predetermined sequence of task operations, each which needs to be processed without interruption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once[2]. According to complexity theory [3], the JSSP is characterized as NP-hard combinatorial optimization problem. Obtaining exact solutions for such problems is computationally intractable [2, 3].Flexible Job Shop Scheduling Problem (FJSSP) is an extension of JSSP which allows an operation to be processed by any machine from a given set [4].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, which is considered one of the most difficult problems in combinatorial optimization [5].

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 [6].

In 1993, Brandimarte[7] was the first man who used tabu search (TS) algorithm about FJSP. After that in 1995Paulli[8] applied hierarchical approach for FJSP. Although the single-objective FJSP has been widely investigated, the research on the multi-objective FJSP is still considered relative limited [9]. In 2002Kacem et al. [10] presented a Pareto optimization approach for multi objective functions. They presented a local approach for solving multi objective optimization problem based on combination of fuzzy logic (FL) and evolutionary algorithm (EAS). In 2004 Rigao[11]developed two heuristics based on tabu search: a hierarchical procedure and a multiple start procedure. In 2005, Xia and Wu[12] presented an optimization and hybrid approach for flexible multi objective FJSP problem. They studied the problem with the hybrid algorithm of the PSO and the simulated annealing(SA). Zhang, Zheng, and Wu [13] 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. Ho et al. [14] studied a hybrid evolution algorithm combined with a guided local search and an external Pareto archive set. Zhang et al. [15] introduced a hybrid algorithm combining PSO algorithm with TS algorithm. Motaghedi-larijani et al. [16]proposed a hybridgenetic

***Corresponding Author:** Samira Zeynali, Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Kerman, Iran. Email: zeynali236@yahoo.com

algorithm with considering the Pareto optimal solutions. They used from a local Search heuristic for improving the chance of obtaining more number of global Pareto-optimal solutions.

In this paper, we study the flexible job shop scheduling problem and three objectives were considered. We have used a Genetic algorithm for generating initial solution and we have also used Tabu search algorithm to improve the quality of solutions.

The paper is organized as follows: Section 2 gives a brief introduction about Flexible job shop scheduling problem and Section 3 describes an overview of Meta heuristic optimization techniques such as Genetic algorithm and Tabu search. And also Section 3 describes the proposed algorithm for Flexible job shop scheduling problem. Section 4 covers numeric example. And finally section 5 deals with the conclusion.

2. Flexible job shop scheduling problem (FJSSP)

Flexible Job Shop Scheduling Problem (FJSSP) has an especial place in industry environments; due to this issue and also due to its mathematical characteristics large number of managers and researchers are considered this problem.

The Flexible job shop scheduling problems consist of n independent jobs $J=\{J1, J2, \ldots, Jn\}$and m Machines $M=\{M_1, M_2, M_3 \ldots M_m\}$. Each job i consists of a sequence of n_i operations O_{i1}, O_{i2}, $O_{i3}\ldots$ O_{ini}. The execution of each operation j of a job i (O_{ij}) requires one machine from a set of given machines called M_{ij}, $M_{ij} \subseteq M$; and the processing time of operation O_{ij} on machine k is P_{ijk}.

During the solving Flexible job shop scheduling problems, the following assumptions are considered:
(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) Jobs and Machines are independent from each other;
(8) There are no precedence constraints among operations of different jobs.
(9) Neither release times nor due dates are specified.

Also in this paper three objectives are considered and all of them will be minimized:

1) Maximum completion time (Makespan);
   Fitness1=Max Cii=1, 2… n n=Number of jobs (1)
   That Ci be the completion time of job Ji

2) The total workload of machines;
   Fitness2= $\sum_{j=1}^{m} Workload_j = \sum_{j=1}^{m} P_{ijk}$ m=Number of machines (2)

3) The critical machine workload;
   Fitness3=$\mathrm{Max}_{j=1,2,\ldots,m}\left[Workload_j\right]$(3)

That the Eq. (1) denotes the maximum finishing time considering all the operations and about Eq. (2) must be clear that the cost of any assignment is directly related to the processing time of the job on the machine. And finally Eq. (3) is equal to the maximum of workloads for all machines. In all of scheduling problems between objectives Makespan has an important and key role. And it is clear that all criteria are conflicted to each other.

3.Proposed Algorithm

In this section the hybrid proposed algorithm to solving the Flexible job shop scheduling problem with considering three objectives will be explained.

There are several intelligent algorithms for solving the NP-hard optimization problems; genetic algorithm, particle swarm optimization, simulated annealing, and ant colony optimization are some of these algorithms. Between these algorithms, Genetic algorithm is one of the most popular meta heuristic algorithms that developed by Holland[17]. which are based on the mechanics of natural selection and genetics to search through decision space for optimal solutions[18].

Generally, performing a genetic algorithm is simple. GA starts with an initial population of possible solutions. For evaluating the solutions in any iteration, an objective function is defined. Then 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[19]. In each new generation by two genetic operators (Crossover and Mutation) Chromosomes are combined or changed and new chromosomes (offspring) are

generated. The new population is then used in the next iteration of the algorithm. Finally, the new solutions are used to replace the poorer of the original solutions and the process is repeated. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population [19].

In the following sections, first the Encoding scheme is considered and then the steps of used Meta heuristic algorithms will be explained.

3.1. Encoding scheme

Since in FJSSP any operation can perform on any machine, in coding the problem we have to consider machines in addition of operations. So according to Zhang et al.[15], for coding the problem, Chromosomes are divided in two sections; in other word one solution is shown by two chromosomes; one chromosome for operations and one chromosome for used machine. The number of Gens in two chromosomes is same and calculated as below:

$$\text{Number of Jens}=\sum_{j=1}^{Numberof Operationi} O_{ij} \qquad \text{i=1, 2, 3… n} \tag{4}$$

To illustrate the issue, consider a problem with four jobs each of which must be performed on five machines (Table 1). Jobs consist of 2, 2, 4, 3 operations respectively. In the next step in order to define the sequence of operations, for each of them, in any chromosome the job index must be repeated as number as its operation, and for generating a Machine chromosome in any Jen a random number is randomly generated between 1 and 5. For example an Operation chromosome and Machine chromosome cab be as follow:

Operation chromosome = [2, 3, 1, 2, 2, 1, 3, 4, 3, 1, 4, 3]

Machine chromosome =[2, 2, 3, 4, 5, 4, 4, 5, 1, 2, 2, 3]

3.2. Genetic operations:

3.2.1. Crossover:

Crossover is the process in which the chromosomes are mixed and matched in a random fashion to produce a pair of new chromosomes(offspring)[20]. In this hybrid algorithm the crossover operations of both Machine and Operation chromosome perform separately.

The Crossover procedure for Machine chromosome is as follow, first 2 chromosomes are chosen randomly and these two chromosomes should not be same to each other. And then for performing Crossover, Two-Point Crossover mechanism is used.

Two-Point Crossover to Machine Chromosome- Two random points is chosen on the individual chromosomes (strings) and the genetic substring is exchanged at these points. For instance, in a previous problem **(**Table 1), two random Machine chromosomes with feasible genes can be as follows:

Parent 1 = [2, 2, 3, 4, 5, 4, 4, 5, 1, 2, 2, 3]
Parent 2 = [3, 2, 3, 5, 4, 5, 5, 4, 2, 1, 1, 2]

The offspring produced from these parents by applying the above-mentioned operators are as follows:
Offspring 1 = [2, 2, 3, 4, 5, 5, 5, 4, 2, 2, 2, 3]
Offspring 2 = [3, 2, 3, 5, 4, 4, 4, 5, 1, 1, 1, 2]

And also this crossover procedure guarantees that the results are feasible.

In order to perform crossover operation on operation chromosome, 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 were move 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 [16]. For instance, in a mentioned problem, two random operation chromosomes with feasible genes can be as follows:

Parent 1 = [2, 3, 4, 2, 3, 4, 1, 3, 1, 3, 2, 1]
Parent 2 = [4, 2, 3, 2, 1, 4, 2, 1, 3, 1, 3, 3]

And two groups of jobs are J1= {1, 4} & J2= {2, 3}; the offspring produced from these parents by applying the above-mentioned crossover procedure are as follows:

Offspring 1 = [2, 3, 4, 2, 2, 4, 1, 3, 1, 3, 3, 1]
Offspring 2 = [4, 2, 3, 2, 4, 1, 2, 1, 3, 1, 3, 3]

3.2.2. Mutation:

Mutation operator is the process used to rearrange the structure of the chromosome to produce a new one [20]. For applying the mutation procedure to both chromosomes, two-point mutation is used. In this way two randomly point of one chromosome is chosen and then the value of Jens is changed between these two points. The offspring produced by applying the above-mentioned Mutation procedure are as follows:
Two-point mutation with random points.

Machine Chromosome:
Parent = [2, 2, 3, 4, 5, 4, 4, 5, 1, 2, 2, 3]
Offspring = [2, 2, 5, 4, 5, 4, 4, 3, 1, 2, 2, 3]

Operation Chromosome:
Parent = [4, 2, 3, 4, 1, 2, 1, 3, 1, 3, 2, 3]
Offspring = [4, 2, 2, 4, 1, 3, 1, 3, 1, 3, 2, 3]

3.3.Tabu Search

To date, TS appears to be one of the most successful and most widely used Meta heuristics, achieving excellent results for a wide variety of problems[21]. The Tabu Search algorithm (TS) was initially proposed by Glover[22]. Tabu search, like other meta-heuristic approaches, is based on the local search procedure. Starting from a given initial solution s_0 (e.g., found by a constructive heuristic algorithm), at each iteration t, the method moves from solutions$_{t-1}$ to the best solution in its neighborhood (i.e., the set of solutions obtained from s_{t-1} by applying all the possible moves), even if this causes a deterioration in the objective function value (in order to escape from the local optimum). To avoid cycling, some solutions of the neighborhood are declared forbidden, or tabu, for a given number of iterations (tabu tenure). The search stops whenever a given stopping rule is satisfied.
An algorithmic outline of a simple TS algorithm is given in below:

1- Generating an initial solution: select an initial solution based on the section.
2- Initializes all the memory structures used during the run of the TS algorithm, at the start of algorithm all memory are empty.
3- Generating the neighborhood solution, in this paper for more speed after that all of neighbor solutions were generated the subset of them are chosen randomly that are called Candid list.

In addition, the neighborhood is defined as the set of solutions obtained from current solution and any neighborhood solution is generated by changing an element of current solution.

4- Evaluating the neighbors and select best solution.
5- Updating the memory structures

3.4.Hybrid GA-TS algorithm

GA is a population-based meta-heuristic that can be used for NP-hard optimization problems. Moreover, TS is a meta-heuristic that is designed for finding a near-optimal solution of combinatorial optimization problems. Therefore, the combination of GA and TS can improved the quality of found solution with compared the GA. The final TS solution is used as one of the initial solutions of the GA algorithm.

The proposed GA-TS hybrid algorithm includes two phases. In the first phase, one solution is generated by GA algorithm and then this solution is improved by TS algorithm.

The general outline of the proposed GA-TS hybrid algorithm for the FJSP is as follows.

Step 1: First, problem data are read and then N random chromosomes are produced as mentioned in section 3.1.
Step 2: For each one of the *N* chromosomes produced, the objective function (Obj$_{(i)}$) is calculated as follows:

$$\text{Obj}_i = w1 * \frac{Makespan(i) - F1Min + \gamma}{F1Max - F1Min + \gamma} + w2 * \frac{TotalWorkload(i) - F2Min + \gamma}{F2Max - F2Min + \gamma} + w3 * \frac{MaxWorkload(i) - F3Min + \gamma}{F3Max - F3Min + \gamma} \quad (5)$$

$F1Min$: Minimum Makespan in population;
$F1Max$: Maximum Makespan in population;
$F2Min$: Minimum Total workload in population;
$F2Max$: Maximum Total workload in population;
$F3Min$: Minimum Max workload in population;
$F3Max$: Maximum Max workload in population;
γ: A little positive value;

W_1:Weight of first objective
W_2:Weight of second objective
W_3:Weight of third objective

In equation (5) the Makespan (i), Total workload (i), Max workload(i) are calculated according to equation (1), (2), (3) respectively. Also if denominators were equal to zero, then it would be impossible to calculate Obji. Then, γis added in the equation (5).

Step 3: Using the Tabu search algorithm for improvement the solutions and for this, first the neighborhood solution from current solution is generated and to prevent of high calculation, a candidate list of neighborhood solution are chosen randomly; and these random solution are compared to each other.

Step 4: Calculation the objective function ($Obj_{(i)}$)to reached solutions as step 2.

Step 5: Selection the best parents of population; in this section first the generated population is sorted ascendant and then pop chromosomes of best parent are chosen for next step.

Step 6:producing offspring chromosomes from parent chromosomes. In this step the Crossover and Mutation procedure that was explained in section 3.2 is used.
It should be mentioned that the mutation rate is decreasing and uses function (6),so that in the final generation, the mutation rate will be zero[19].

$F_{Murate}=1-\frac{Ng}{G}$ (6)

Where *G* is the number of generations and *Ng* is the generation number index.

Step 7: Repeat steps 4-6 until the chromosomes do not change from one generation to the next.

The procedure of the hybrid algorithm is summarized in Fig .1.

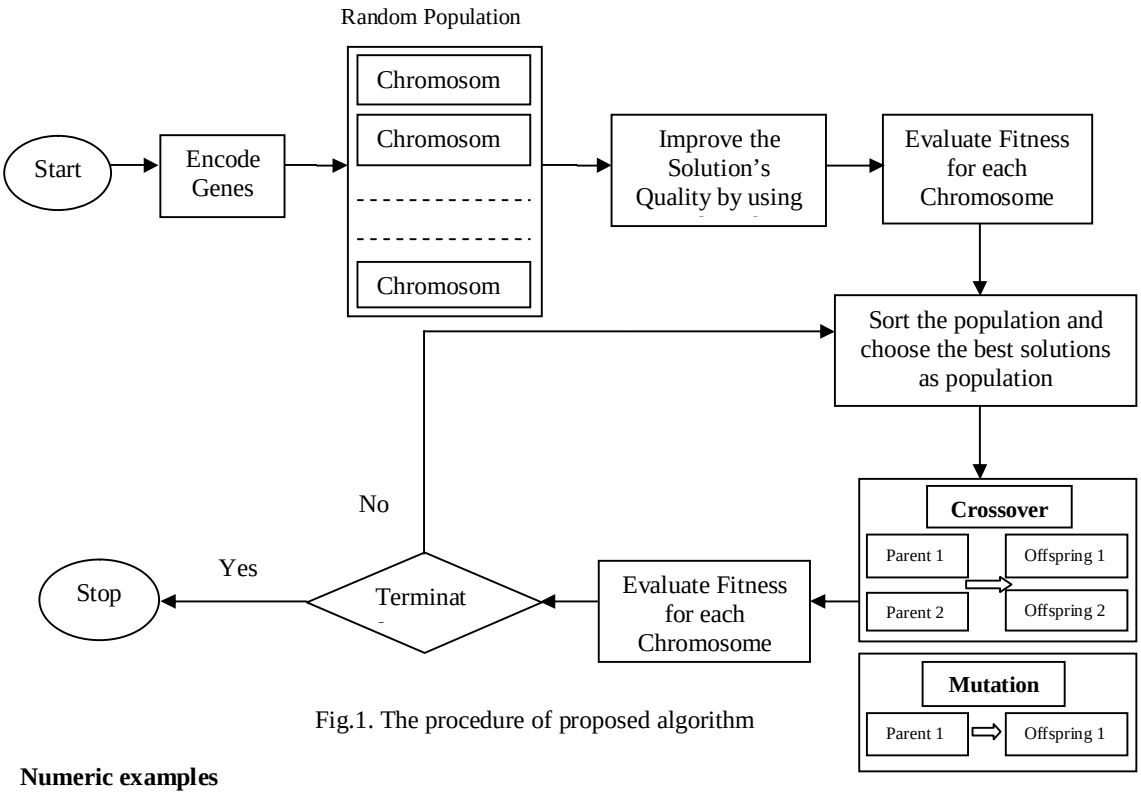

Fig.1. The procedure of proposed algorithm

4. Numeric examples

In this section in order to explain the function of proposed algorithm, we compared the algorithm with other papers and for this issue two kind of instances (small scale problem with 4 jobs and 5 machines and middle scale with 10 jobs and 10 machines) are considered same as table 1, 3 respectively.

At first because of increasing the efficiency of algorithm the best values for parameters were calculated. We analyzed the solutions by considering 3 levels for POP size (Pop=150, 200, 300), 3 levels for Crossover

rate (0.7, 0.8, 0.85) and 2 levels for Mutation rate (0.1, 0.2) and for each combination of parameters the algorithm was run for 10 times. The result of analyzing the reach solutions by ANOVA is shown in the Fig 2and as it is clear in the Fig 2 the best values for parameters are pop size=300, Crossover rate=0.8, Mutation rate=0.2.

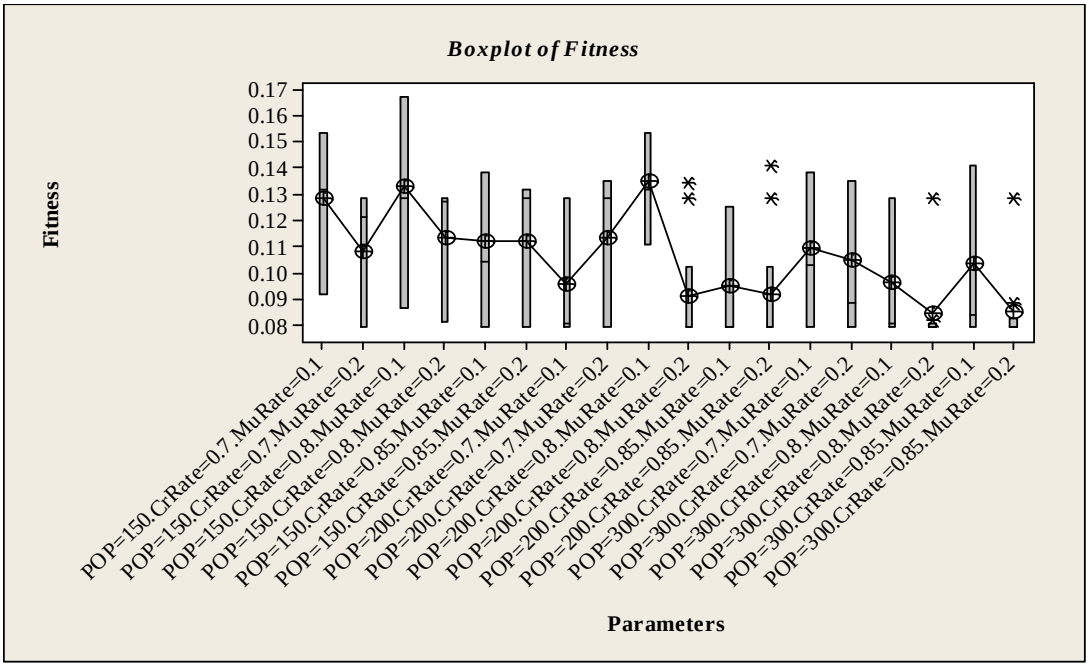

Fig.2. Analyze of Parameters

Example 4.1
This instance is taken from[16] and is considered as a small scale instance.

Job	Operation	M_1	M_2	M_3	M_4	M_5
J_1	O1,1	2	5	4	1	2
	O1,2	5	4	5	7	5
	O1,3	4	5	5	4	5
J_2	O2,1	2	5	4	7	8
	O2,2	5	6	9	8	5
	O2,3	4	5	4	54	5
J_3	O3,1	9	8	6	7	9
	O3,2	6	1	2	5	4
	O3,3	2	5	4	2	4
	O3,4	4	5	2	1	5
J_4	O4,1	1	5	2	4	3
	O4,2	5	1	2	1	2

Table 1. The processing times of jobs on the machines

The best found solution in related paper and proposed algorithm are as table 2. And as is clear the proposed algorithm finds one good solution more.

Objectives	Fattahi et al.		Proposed algorithm		
Makespan	13	12	13	12	11
Total Workload	33	32	33	32	32
Max Workload	7	8	7	8	10

Table 2. Comparison results on small scale instance

Example 4.2 In this example, the efficiency of algorithm on Middle scale problems is tested. The table 3 is shown the processing time of middle scale problem that is taken from [10].

Job	Operation	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
J_1	O1,1	1	4	6	9	3	5	2	8	9	5
	O1,2	4	1	1	3	4	8	10	4	11	4
	O1,3	3	2	5	1	5	6	9	5	10	3
J_2	O2,1	2	10	4	5	9	8	4	15	8	4
	O2,2	4	8	7	1	9	6	1	10	7	1
	O2,3	6	11	2	7	5	3	5	14	9	2
J_3	O3,1	8	5	8	9	4	3	5	3	8	1
	O3,2	9	3	6	1	2	6	4	1	7	2
	O3,3	7	1	8	5	4	9	1	2	3	4
J_4	O4,1	5	10	6	4	9	5	1	7	1	6
	O4,2	4	2	3	8	7	4	6	9	8	4
	O4,3	7	3	12	1	6	5	8	3	5	2
J_5	O5,1	7	10	4	5	6	3	5	15	2	6
	O5,2	5	6	3	9	8	2	8	6	1	7
	O5,3	6	1	4	1	10	4	3	11	13	9
J_6	O6,1	8	9	10	8	4	2	7	8	3	10
	O6,2	7	3	12	5	4	3	6	9	2	15
	O6,3	4	7	3	6	3	4	1	5	1	11
J_7	O7,1	1	7	8	3	4	9	4	13	10	7
	O7,2	3	8	1	2	3	6	11	2	13	3
	O7,3	5	4	2	1	2	1	8	14	5	7
J_8	O8,1	5	7	11	3	2	9	8	5	12	8
	O8,2	8	3	10	7	5	13	4	6	8	4
	O8,3	6	2	13	5	4	3	5	7	9	5
J_9	O9,1	3	9	1	3	8	1	6	7	5	4
	O9,2	4	6	2	5	7	3	1	9	6	7
	O9,3	8	5	4	8	6	1	2	3	10	12
J_{10}	O10,1	4	3	1	6	7	1	2	6	20	6
	O10,2	3	1	8	1	9	4	1	4	17	15
	O10,3	9	2	4	2	3	5	2	4	10	23

Table 3. The processing times of jobs on the machines

The best found solution in related paper and proposed algorithm are as table 4.

Objective	Heuristic method (SPT)	Classic GA	Kacem's Approach	GA+TS		
t_M	16	7	7	8	8	9
W_T	59	53	45	44	46	45
W_M	16	7	6	7	6	7

Table 4. Comparison results on middle scale instance

5. CONCLUSION

The Flexible Job Shop Scheduling Problem is one of the most difficult problems in scheduling environments and many researchers have analyzed this problem. In current paper the FJSP is considered and one Meta heuristic algorithm based on genetic algorithm was proposed for it. Also for improve the quality of solutions the genetic algorithm was combined by Tabu search method. In this paper, our goal is to find an optimum sequence for minimizing the fitness function and for calculating the fitness function three objective functions were calculated. Makespan, Total Workload of machines and maxim of Machine workloads and all of them should be minimized. We coded this algorithm in VBA and the reached solutions of algorithm in two kinds of instances (small and large scales) were compared with other papers in final section.

REFERENCES

[1]E.Baghal Azardoost and N.Imanipour, 2011.A Hybrid Algorithm for Multi Objective Flexible Job Shop Scheduling Problem. Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management Kuala Lumpur, Malaysia, January 22 – 24.

[2]M. Ventresca andB. M. Ombuki, 2004.Ant Colony Optimization for Job Shop Scheduling Problem. Brock University, Technical Report CS-04-04

[3]M.R.Garey and D.S, 1979.Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H Freeman and Company.

[4]Li-Ning Xing, Ying-Wu Chen and Ke-Wei Yang, 2011. Multi-population interactive co evolutionary algorithm for flexible job shop scheduling problems. Comput Optim Appl 48: 139–155DOI 10.1007/s10589-009-9244-7

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

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

[7]P.Brandimarte, 1993. Routing and scheduling in a flexible job shop by tabu search. Annals of Operation Research,41, 157-183.

[8]J.Paulli, 1995.AHierarchical Approach for the FMSScheduling Problem. European Journal of Operational Research Vol. 86, pp. 32-42.

[9]J. Li, Q. Pan, S. Xie and S. Wang, 2011. A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problems, Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844Vol. VI, No. 2 (June), pp. 286-296

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

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

[12]Weijun, Xia.et.al. 2005.An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computers and Industrial Engineering, 48,409-425.

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

[14]N. B. Ho and J. C. Tay,2008. Solving Multiple-Objective Flexible Job Shop Problems by Evolution and Local Search. IEEE Transactions on Systems, Man, and Cybernetics, Part C:Applications and Reviews, Vol. 38(5), pp. 674-685.

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

[16]A.Motaghedi-larijani, A.Sabri-laghaie and M.Heydari,2010.Solving Flexible Job Shop Scheduling with Multi Objective Approach. International Journal off Industrial Engineering & Production Research, Volume 21, Number 4 pp.. 197-209

[17]J. H. Holland, 1975. Adaptation in Natural and Artificial Systems, University of Michigan.

[18]Chung-Wei Feng; Liang Liu, Scott A. Burns 1997.Using Genetic algorithms to solve construction Time-Cost Trade-off problems. Journal of computing in civil engineering.

[19]N.Shahsavari Pour, M.Modarres, M.B.Aryanejad and R.Tavakoli Moghadam, 2010.The Discrete Time-Cost-Quality Trade-off Problem Using a Novel Hybrid Genetic Algorithm. Applied Mathematical Sciences, Vol. 4, no. 42, 2081 – 2094

[20]M. Adel El-Baz, 2004. A genetic algorithm for facility layout problems of different manufacturing environments, Computers & Industrial Engineering 47, 233–246

[21]M .Dorigo and T. Stutzle, 2004.Ant Colony Optimization. MTI Press, Page 50.

[22]F.Glover, 1989. Tabu search—Part I. ORSA Journal on Computing, 1(3), 190–206.