Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management
Istanbul, Turkey, July 3 – 6, 2012

Genetic Algorithm based on Some Heuristic Rules for Job Shop Scheduling Problem

Amer A. Boushaala, Mohamed A. Shouman and Somaia Esheem
Department of Industrial Engineering and Manufacturing Systems
Garyounis University, Benghazi, Libya

Abstract

Job-shop scheduling problem (JSSP) is one of the most difficult scheduling problems, as it is classified as NP-hard problem. In this paper, a hybrid approach based on a genetic algorithm and some heuristic rules for solving (JSSP) is presented. The scheduling heuristic rules are integrated into the process of genetic evolution. The algorithm is designed and tested for the scheduling process in two cases in which the first generation the initial population is either random generation or the results obtained from some active heuristics rules. To speed up the generation of heuristics rules, ~~a~~ weighted priority rules are used as heuristic rules for achieving better performances for generating feasible schedules. The results of the purposed hybrid algorithm of this paper are promising where these results are compared to benchmark problems results.

Keywords
Genetic algorithms, job-shop scheduling, hybrid systems, heuristic rules.

1. Introduction
Scheduling is a well-known problem that deals with the efficient allocation of resources with respect to time in order to perform a collection of tasks. Job shop scheduling is one of the widely studied and most complex combinatorial optimization problems, as it is classified as NP-Hard problem and therefore no deterministic algorithms can solve them in a reasonable amount of time [1]. Job shop scheduling was primarily treated by the branch-and bound method [2], and some heuristic procedures based on priority rules. In the recent years, many researchers have become interested in such problems. Many complex problems have been studied and solved by met heuristic search techniques such as Tabu Search (TS), Genetic Algorithm (GA) [3, 4, and 1] and Simulated Annealing (SA). These techniques have been employed to deal with complex scheduling problems which are capable of producing high quality solutions with a reasonable computational effort [5]. Genetic algorithm proposed by Holland (1975) and Goldberg (1989) have been applied in a number of fields, e.g. mathematics, engineering, biology, and social science as mention by Zandieh et *a*l [6]. GA is an optimization search method motivated by natural selection and natural evolution. It maintains a population where each individual is characterized by its chromosome. Each chromosome consists of a sequence of genes. The idea is to get a new population with better desired characteristics than the previous. The process is repeated for the newly obtained set of solutions until a desired level of performance is obtained. Genetic algorithms have proven their efficiency in solving high complexity combinatorial optimization problems, thus many researchers have applied them also to the scheduling problems [7, 3, and 8]. The implementation time of the GA can be defined as the time required by the algorithm to render an optimal or satisfactory solution. This time reflects the solution quality comprising each generation. If the quality of solutions is poor, i.e. the individuals are beyond the fitness function or imposed constraints, then the results seem to be hopeful but, the GA will take more time to render or reach the best solution. Some techniques and operators are used to improve the solutions quality. The performance of GAs solutions depends on the quality of the initial population [9-12] on which the quality and performance of the next populations generations will depend ~~on~~. This article focuses on this fact and tries to use some recommended heuristics and developed heuristic rules in the current article for generating good solutions for JSSP and then used these generated solutions rendered by heuristics as initial population for the genetic algorithm. The remainder of the paper is organized as follows. Section 2 describes problem structure. Sections 3, 4 present the optimization approach for the JSSP. Section 5 tests the optimization approach on benchmark problem. Finally, conclusion and remarks are given in Section 6.

2. Job-Shop Scheduling Problem

JSSP can be stated as a set of n jobs to be processed on a set of m machines, where each job *j* visits a number of machines in a predetermined order. The processing times for each job at each machine are given and no machine can process more than one job at a time. If a job is started on a machine, then it cannot be interrupted [6]. The problem is finding a schedule of the jobs on the machines. The assumptions of the present problem are:

- Every job has a unique sequence on *m* machines. There are no alternate routings.
- There is only one machine of each type in the shop.
- Processing times for all jobs are known and constant.
- All jobs are available for processing at time zero.
- Machine absences are not allowed.
- Transportation time between machines is zero.
- Each machine can perform only one operation at a time on any job.
- An operation of a job can be performed by only one machine.
- Operation cannot be interrupted.
- A job does not visit the same machine twice.
- An operation of a job cannot be performed until its preceding operations are completed.
- Each machine is continuously available for production.
- There is no restriction on queue length for any machine.
- There are no limiting resources other than machines/workstations
- The machines are not identical and perform different operations [6].

3. Generating Dispatching Rules

3.1 Common priority rules

Priority dispatching rules are actually the most widely used for solving JSSP where all the operations available to be scheduled are assigned a priority. The operation with the highest priority is chosen to be sequenced first. The heuristic rules used in the current work and of promising results as indicated by many researchers and of a significant optimization capacity are listed in Table 1 [3, 1, and 13].

Table 1. Common priority rules

Expression	Description
Shortest Processing Time (SPT)	The job with shortest time on machines selected. $P_i \leq P_{i+1} \leq P_{i+2} \leq \ldots\ldots\ldots\ldots \leq P_n$
Longest Processing Time (LPT)	The job with longest processing time on machine is selected. $P_i \geq P_{i+1} \geq P_{i+2} \geq \ldots\ldots\ldots\ldots \geq P_n$
Minimum Slack Time Per Operation (MINSOP)	Time remaining until the due date - processing time remaining
Minimum Due Date (MINDD)	The job with the earliest due dates processed first. $D_i \leq D_{i+1} \leq D_{i+2} \leq \ldots\ldots\ldots\ldots \leq D_n$
RANDOM (random selection)	Selects the next job to be processed at random.
Critical Ratio (CR)	Remaining due date / Remaining processing time
Most work remaining (MWKR)	select the operation associated with the job of the most work remaining to be processed.

Least work remaining (LWKR)	select the operation associated with the job of the least work remaining to be processed
Most Operation Remaining (MOPNR)	select the operation that has largest number of successor. operations.
Shortest Remaining Minimum Processing Time (SRMPT)	Min (processing time remaining – min processing time).
Longest Remaining Maximum Processing Time (SRMPT)	Max (processing time remaining – max processing time).
Fewest Number of Operation Remaining (FOPNR)	Min Ratio (operation remaining for job / sum of operations).
Greatest Number of Operation Remaining (GOPNR)	Max Ratio (operation remaining for job / sum of operations).
SPT/WKR	(smallest weight ratio of processing time to work remaining)
Shortest Weight Process Time (SWPT)	$\frac{P_j}{W_j} \leq \frac{P_{j+1}}{W_{j+1}}$
Longest Weight Process Time (LWPT)	$\frac{P_j}{W_j} \geq \frac{P_{j+1}}{W_{j+1}}$
Shortest Weight Mean Processing Time (SWMPT)	$\frac{P_k}{W_k} \leq \frac{P_{k+1}}{W_{k+1}}$
Longest Weight Mean Processing Time (LWMPT)	$\frac{P_k}{W_k} \geq \frac{P_{k+1}}{W_{k+1}}$

3. 2 Developed priority rules

Nine priority rules are proposed in the current work and tested on seventy instance cases representing job shop problem. Fifty percent of these seventy cases are classical job shop problem while the other fifty percent are flexible job shop problem. Also the previous eighteen priority rules are used for scheduling the instance cases under consideration. The rendered results on the instance cases under consideration by either the common priority rules (eighteen) or the new proposed priority rules are compared under six measuring performance criteria. These measuring performance criteria are makespan, lateness, number of late/tardy jobs, tardiness, earliness, and number of early jobs. Also two scenario conditions are considered for the experimental study. These scenarios are based on the considered weight of each job where this weight is based on either the significant importance or based on its relative time to other times. Based on this assumption the first experimental scenario has been done based on randomized weight values for the jobs to be scheduled while the second scenario is based on relative time of the job to other jobs times. These two scenarios are considered for the experimental work in this study and compared to see the influence of weight on the scheduling process. However, the mathematical equations for the proposed rules are listed as follows:

- Minimum Weighted Due Date (MINWDD): for this rule, the job with the earliest weighted due date is processed first. $W_iDi_{\leq} W_{I+1} D_{i+1} \leq W_{i+2} D_{i+2} \leq \ldots\ldots \leq W_nD_n$

- Weighted Shortest Processing Time (WSPT): for this rule, job with weighted shortest processing time on machine is selected. $W_iP_i \leq W_{i+1}\,P_{i+1} \leq W_{i+2}\,P_{i+2} \leq \ldots\ldots \leq W_nP_n$
- Weighted Longest Processing Time (WLPT): for this rule, the job with weighted longest processing time on machine is selected. $W_iP_i \geq W_{i+1}\,P_{i+1} \geq W_{i+2}\,P_{i+2} \geq \ldots\ldots\ldots \geq W_nP_n$
- Minimum Weighted Slack Time Per Operation (MINWSOP): The weighted time remaining until the due date minus weighted processing time remaining.
- Weighted Critical Ratio (WCR):

  $$WCR = \frac{\text{Remaining due date}}{\text{Remaining processing time}} \times \text{weight}$$

  $W_iCR_i \leq W_{i+1}CR_{i+1} \leq W_{i+2}CR_{i+2} \leq \ldots\ldots\ldots \leq W_nCR_n$

  The job with the minimum value of WCR is scheduled next .This rule calculates the weighted ratio of demand time to supply time. When the weighted ratio exceeds a value of 1, there is sufficient time a available to complete The job if the queue times are properly managed. If the weighted ratio is less than 1, the job will be late unless processing times can be compressed. The WCR rule has slightly more intuitive appeal, since the WCR itself has a precise meaning.
- Fewest Weighted Number of Operation Remaining (FWOPNR).
- Greatest Weighted Number of Operation Remaining (GWOPNR).
- Most Weighted Work Remaining (MWWKR): the operation associated with the job having the most weighted work remaining to be processed.
- Least Weighted Work Remaining (LWWKR): the operation associated with the job having the least weighted work remaining to be processed.

The twenty seven rules are tested on seventy case instances of benchmark problems. The experiments are performed under two scenarios. The results clearly show that the developed heuristic rules are very promising and achieved better results than the most common and recommended heuristics used for many different instances [14]. The problem instances are tested through a software program that is designed and built in a structure of a decision support system. This system is capable of running all the twenty seven rules for any problem in the aim of achieving the minimum make span or the best solution that can be obtained by these embedded 27 rules for the tested or considered problem. The resulted solutions from the common priority rules (eighteen) and the developed priority rules are considered the first generation the initial population of the presented genetic algorithm.

4. Genetic Algorithm Features

In this research a hybrid genetic algorithm (HGA) based on rendered solutions as a first population for the genetic algorithm is presented. Figure 1 presents the flow chart of the proposed algorithm. The next sections introduce the genetic algorithm features.

4.1 Chromosome representation

The representation of a chromosome follows the genetic representation as given by [15] where each gene corresponds to one of *n* jobs and a chromosome corresponds to a solution (a sequence of jobs).The chromosome length is the sum of all the operations on all jobs. A string of triples (J, I, M) is used, one for each operation, where J is the considered job, I is the progressive number of that operation within job j, and M is the machine assigned to that operation. Table 2 presents one of the benchmark problems used in testing the genetic algorithm [3]. For such scenario, the chromosome representation is given in Figure 2. Each chromosome is represented by a list of job order, respective operation and machine sequence. In figure 2, the chromosome consists of five jobs each of five operations. The order of the job to be done is the first job, job $_4$ followed by Job_5, Job_2, Job_3 and Job_1. In Job_4, the first operation (i.e. O_1) is to be done at machine Mc_2, then followed by operation two (i.e. O_2) at machine Mc_1,then followed by O_3 at machine Mc_5, then followed by O_4 at machine Mc_4, then followed by O_5 at machine Mc_3. Similarly, for job number 5 (i.e. Job_5), the first operation is to be done at Mc_2 and the second operation is to be done at Mc_4, then followed by O_3 at machine Mc_5**,** then followed by O_4 at machine Mc_1**,** then followed by O_5 at machine Mc_3. The same procedure is applied for the other jobs**.**

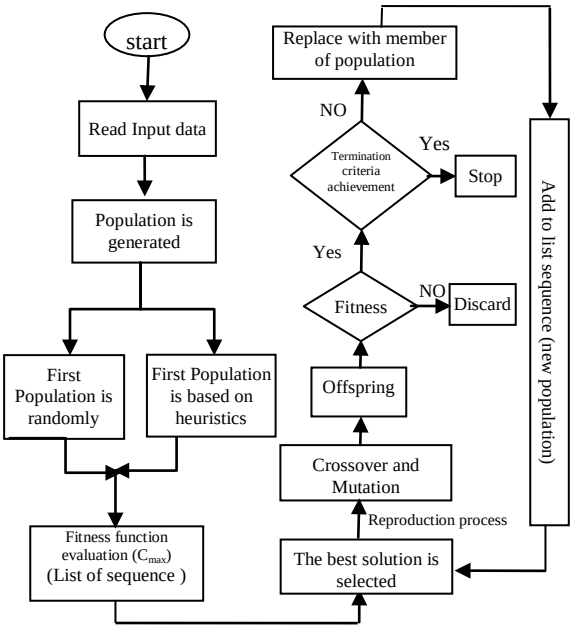

Figure 1: Proposed algorithm flow chart

Table 2. Machine order and processing time

Job	Operation				
	1	2	3	4	5
1	3,1	1,3	2,6	4,7	5,6
2	2,8	3,5	5,4	4,4	1,1
3	3,5	4,2	5,3	2,1	1,2
4	2,5	1,4	3,2	4,3	5,1
5	2,3	4,3	5,4	1,4	3,1

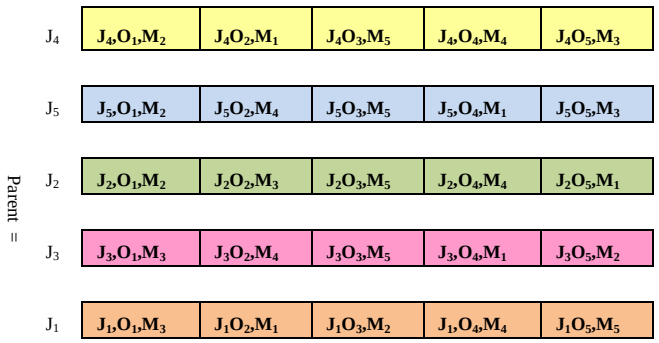

Figure 2: Chromosome representation of solution

4.2 Initial population generation

The first population currently set to (27) solutions is generated in two scenarios, in the first one the initial population is randomly generated for each instance based on JSP domain search space. In second scenario, the initial population is generated based on the results obtained of both considered twenty seven heuristics rules. This allows

of the capability of the proposed genetic algorithm (GAs) for vastly obtaining a good initial set of feasible schedules (individuals). All the solutions in first population have not the same sequence order in order to maintain the diversity of individuals.

4.3 Evaluation function

The evaluation function is a measure of the fitness and quality of a chromosome relative to the other chromosome in a population. For a job shop, the make span (C_{max}) is a good way to determine machine utilization. The chromosome with the least fitness function value is the most desired or considered as the best rendered solution.

4.4 Selection strategy

The proposed genetic algorithm uses a ranking strategy for chromosome selection; the population is ranked according to the fitness values of its members. Two individuals based on minimum make span are selected from the population for reproduction process. This is a process used to determine the number of trials for one particular individual used in reproduction. This means that all individuals in the population have the chances to reproduce. In each generation, a selection scheme is used to select the survivors to the next generation according to their fitness values.

4.5 Crossover operators

Offspring of crossover should represent solutions that combine substructures of their parental solutions. Many various crossover operators can be used such as single-point crossover ,two-point crossover, partial - mapped crossover (PMX),order crossover (OX), cycle crossover (CX), edge combination crossover (ERX) and job based order crossover etc. In the current research two methods of crossover operators are applied, single point crossover and multi-point crossover. Both the applied crossover operators are selected randomly for each instance and remain constant. After the crossover is done, the new part chromosomes are checked for the validity of being new comer. If not then, a repairing action is performed. This process is done to assure that the resulting string presents a complete sequence for processing the jobs.

4.6 Mutation operators

To maintain the genetic diversity from one generation to the next, the offspring solutions are mutated. In general, in this operation two genes are selected at random and their positions are exchanged. The need of mutation is to ensure that no item/task occurs more than once in the chromosome. Every mutation operator is applied to one chromosome and results in a different chromosome. Several mutation operators have been proposed for permutation representations, such as inversion, insertion, displacement, reciprocal exchange mutation, and shift mutation. The reciprocal exchange mutation is applied for the GA with probability 0.3 in this paper.

4.7 Termination criterion

The most common method is to stop after a predetermined number of generations have been evaluated. Another commonly used method is to terminate when the solution crosses a desired threshold or fitness value. In this paper 200 generations are used as stopping criteria where it was a balance level and no more enhancements can be achieved. The final and the most important step in the proposed algorithm is to choose the individual to be replaced by *child*. In this paper, if the child has the same fitness value with the member of population it is considered as the worst individual in the population. In this case and instead of dropping that child, it is replaced by the old one to given another chance for the old individual to reproduce. However, if the makespan for the child is less than the worst one and not equal to any member of population, replace the worst individual with child'. If there is a member having the same makespan value, then replace it by the member with the child [16]**.** To test the performance of the proposed algorithms, 70 benchmark problems are tested and compared with several heuristic rules and proposed rules. The problems are of 5×5 ($N \times M$ represents N jobs and M machines).

5. Computation Results

The proposed algorithm is applied on benchmark problems to determine its effectiveness. Two scenarios are considered for the implementation process as cited before. In order to compare the performance between the HGA in two scenarios and heuristic rules, both approaches on several job shop scheduling problems are applied. The results of the implementation process for the two scenarios are shown in Table 4. In this table column 1 presents recommended heuristic rules while column 2 presents developed heuristic rules. Column 3 presents the results based

on GA of first generation rendered by heuristics while column 4 presents the results of GA of first generation of random selection. The best solutions are indicated from the resulted data and it is found by the GA based on both the recommended and developed heuristic rules. Also these rendered results of GA based on both the recommended and developed heuristic rules are founded to be the best of all the implemented scenarios and cases. It achieves the best for 62 instances of all the seventy considered instances (89%), compared to 66% when applying GA based on randomization population from the domain search space. In case of applying only the developed heuristic rules the best solution are obtained by 40% while it is 37% when applying the recommended heuristic rules. Figure.2. exhibits comparison measures for the considered cases and the degrees of optimality of obtained solutions and the robustness of the proposed GA in the current research.

Table 4. The comparison results between different approaches

P	1	2	3	4	P	1	2	3	4
1	31	31	29	30	36	32	30	30	30
2	44	45	38	38	37	37	35	35	36
3	23	23	23	23	38	23	23	23	23
4	34	35	33	33	39	19	19	18	18
5	37	39	37	37	40	19	19	19	23
6	36	37	35	36	41	26	26	26	28
7	45	47	40	41	42	24	24	22	22
8	19	23	19	20	43	27	27	27	27
9	35	37	35	32	44	31	33	30	30
10	40	40	40	39	45	25	25	25	25
11	35	38	31	30	46	28	31	26	27
12	28	32	27	27	47	39	39	39	39
13	29	29	29	29	48	38	38	38	41
14	33	33	31	31	49	37	38	37	39
15	41	44	41	41	50	41	36	36	36
16	41	41	41	41	51	79	79	54	55
17	26	32	26	26	52	56	56	56	56
18	33	33	33	33	53	31	31	31	31
19	24	22	22	24	54	30	28	23	27
20	32	32	27	32	55	15	15	15	15
21	28	31	28	29	56	29	30	29	29
22	45	44	44	43	57	31	31	31	31
23	26	26	25	26	58	24	26	21	25
24	36	36	36	34	59	27	25	25	25
25	28	27	27	26	60	29	29	28	27
26	43	51	39	43	61	29	26	26	29
27	38	37	37	38	62	23	23	22	22
28	38	38	38	38	63	32	31	31	31
29	63	63	57	57	64	26	27	26	26
30	14	14	13	14	65	42	42	42	40
31	38	46	38	41	66	27	24	24	24
32	37	38	37	42	67	28	27	24	24
33	25	25	25	25	68	46	49	46	46
34	23	23	23	24	69	52	47	47	50
35	23	22	22	21	70	52	47	47	50

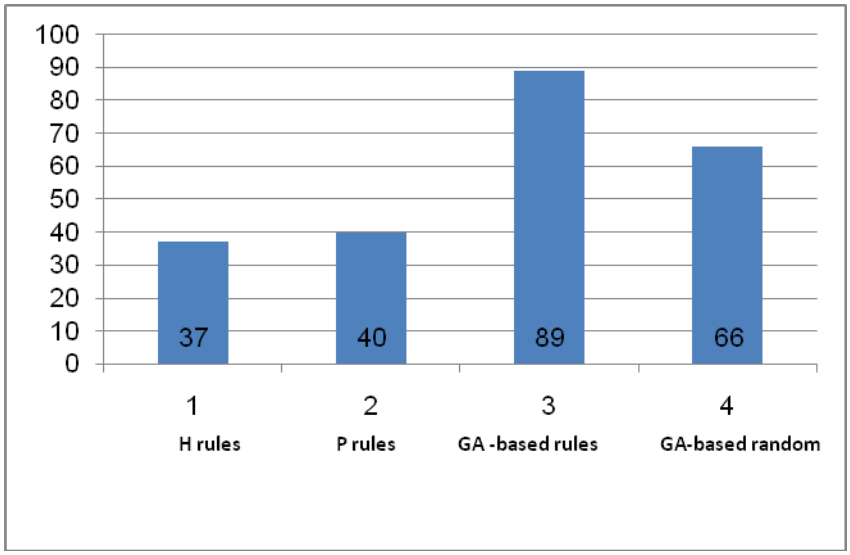

Figure 2: The comparison results obtained in Table 4

6. Conclusions

A hybrid algorithm model based on an integration of a genetic algorithm and some developed and recommended heuristic rules is proposed in this paper. The main objective of this research is to explore more effective and efficient approach or tool for solving scheduling problems in job shops. The most prominent feature of the presented algorithm is a combination of heuristic rules with genetic operators to generate new individuals, which can effectively guide and speed up the genetic search. Different scenarios and cases are considered for evaluation process. The best results have been achieved when integrating the proposed genetic algorithm with the developed and recommended heuristic rules. Other achievements have been obtained in the other studied cases and scenarios but with little value of the fitness function value.

References

1. Kuczapski , M., Micea1 , V., Maniu , A., Cretu , A., 2010, " Efficient Generation of Near Optimal Initial Population to Enhance Genetic Algorithm for Job-Shop Scheduling", ISSN 1392 – 124X information technology and control, Vol.39, No.1.
2. Dorndorf, U. , Pesch, E. , and Phan-Huy,T., 2001 , " Solving the Problem of Open Shop Scheduling" , Journal of Scheduling, Vol. 4, pp. 157 – 174.
3. Rafinejad, S., Ramtin , F., Arabani , A., 2009, "A New Approach to Generate Rules in Genetic Algorithm Solution to a Job Shop Schedule by Fuzzy Clustering", Proceedings of the World Congress on Engineering and Computer Science Vol. II, San Francisco, USA.
4. Omar, M., Baharum , A., Abu Hasan, Y., 2006, " A Job–Shop Scheduling Problem (JSSP) Using Genetic Algorithm-SHOP", Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, statistics and applications University Sains Malaysia , penang, june ,PP. 13-15.
5. Norozi, A., Ariffin, N., and Ismai l., N., 2010, "Application of Intelligence Based Genetic Algorithm for Job Sequencing Problem on Assembly Line ", ISSN 1941-7020, American J. of Engineering and Applied Sciences 3 (1): 15-24.
6. Zandieh, M., Mahdavi, I., and Bagheri, A., 2008, "Solving the Flexible Job- Shop Problem by a Genetic Algorithm", Journal of Applied, 8(24); PP.4650- 4655.
7. Tamilarasi, A., kumar, T, 2010, "An Enhanced Genetic Algorithm with Simulated Annealing for Job-Shop Scheduling", International Journal of Engineering, Science and Technology Vol. 2, No.1, pp.144-151.
8. Kemppainen, K., 2005, "Priority Scheduling Revisited Dominant Rules, Open Protocols, and Integrated Order Management", Helsinki School of Economics.
9. Jain, A., Meeran, S., 1999, "Deterministic Job-Shop scheduling: past, present and future", European Journal of Operational Research, 113 (2), 390−434.

10. Ho, A., Tay, J., 2005, “Evolving Dispatching Rules for solving the Flexible Job-Shop Problem”, IEEE Congress on Evolutionary Computation, 3.
11. Pinedo, M., 2006, “Planning and Scheduling in Manufacturing and Services”, 1st Edition, Springer New York.
12. Lin, S., Goodman, E., Punch, W., 2003, “A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems”, Proceedings of the Seventh International Conference on Genetic Algorithms, PP. 481−488.
13. Huiyuan , R., Lili ,J., Xiaoying , X. , Muzhi , L., 2009, “Heuristic Optimization for Dual-Resource Constrained Job-Shop Scheduling”, International Asia Conference on Informatics in Control, Automation and Robotics, pp.485-488.
14. Boushaala, A. A., Shouman, M. A., and Esheem , S., 2012, “Some Heuristic Rules for Job Shop Scheduling Problem”, accepted for publication in Engineering Journal of Alexandria, EJA .
15. Moghaddas , R., and Houshmand , M., 2008, “Job-Shop Scheduling Problem with Sequence Dependent Setup Times”, Proceedings of the International Multi Conference of Engineers and Computer Scientists 19-21 March, Vol. II, Hong Kong.
16. Lei ,D., 2010, "Solving fuzzy job shop scheduling problems using random key genetic algorithm", Springer-Verlag London Limited Int. J. Adv. Manuf. Technol.