A HYBRID GENETIC ALGORITHM FOR JOB SHOP SCHEDULING PROBLEMS

ISSN 2394–7799 IJAETCS (2016) Vol.3, No.1, 6-17
Research Article

International Journal of Advancement in Engineering, Technology and Computer Sciences

A HYBRID GENETIC ALGORITHM FOR JOB SHOP SCHEDULING PROBLEMS

I. M. El-Desoky [I], *M. A. El-Shorbagy*[I], *S. M. Nasr*[I], *Z. M. Hendawy*[I], *A. A. Mousa*[I,II]

[I]*Department of Basic Engineering Science, Faculty of Engineering, Shebin El-Kom,Menoufia University, Egypt(sarah.nasr.eid@gmail.com).*
[II]*Department of Mathematics, Faculty of Sciences, Taif University, Saudi Arabia.*

ABSTRACT: Job Shop Scheduling Problem (JSSP) is an optimization problem in which ideal jobs are assigned to resources at particular times. In recent years many attempts have been made at the solution of this problem using a various range of tools and techniques. This paper presents hybrid genetic algorithm (HGA) for JSSP. The hybrid algorithm is a combination between genetic algorithm (GA) and local search. Firstly, a new initialization method is proposed. A modified crossover and mutation operators are used. Secondly, local search based on the neighborhood structure is applied in the GA result. Finally, the approach is tested on a set of standard instances taken from the literature. The computation results have validated the effectiveness of the proposed algorithm.

KEYWORDS: Genetic algorithm, Job shop scheduling problem, Local search.

---

1. INTRODUCTION: Scheduling has become a critical factor in many job shops especially for real-world industrial applications [(1)-(3)]. Finding scheduling to achieve the work in a minimum time and more efficiently is called JSSP. The JSSP can be described in the following way: we are given a set of jobs and a set of machines. Each machine handles, at most, one job at a time. Each job consists of a chain of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. The purpose is to find a schedule, that is, an allocation of the operations to time intervals on the machines, that has a minimum duration required to complete[(4)]. The JSSP is among the hardest combinatorial problems. Not only is it complicated, but it is one of the worst NP-complete class members. In general, scheduling problems are NP; NP stands for non-deterministic polynomial, which means that it is not possible to solve an arbitrary instance in polynomial time. So, the JSSP has garnered attention due to both its practical importance and its solution complexity [(5), (6)].

At present, the method for the JSSP mainly includes two kinds, one of which is exact methods and the other approximation methods. Exact methods, such as branch and bound, linear programming and decomposition methods, guarantee global convergence and have been successful in solving small instances[(7)]. In manufacturing systems, most scheduling problems are very complex in nature and very complicated to be solved by exact methods it becomes increasingly important to explore ways of obtaining better schedules that include priority dispatch, shifting bottleneck approach, local search, and heuristic methods. Recently, number of high-level strategies is used to guide other heuristics, known as meta-heuristics led to better and more appreciated results [(8) -(16)]. Therefore, a number of meta-heuristics were proposed in literature for JSSP such as GA [(17) -(18)], simulated annealing (SA) [(19), (20)],ant colony optimization (ACO) [(21)-(23)],tabu search (TS) [(24),(25)], particle swarm optimization (PSO) [(26),(27)], and Consultant Guided Search algorithm (CGS) [(28)].

One of meta-heuristics methods is the GA. GA inspired by the process of Darwinian evolution, has been recognized as a general search strategy and an optimization method which is often useful for attacking combinational problems. In contrast to other meta-heuristics methods the GA utilizes a population of solutions in its search that is not easy to fall into local minima. GA has been used with increasing frequency to address scheduling problems. In [(29)] Lazár introduced a review of frequent approaches and methods for JSSP which most commonly are used in solving this problem. From this review we can say that GA is an effective metaheuristic to solve combinatorial optimization problems, and has been successfully adopted to solve the JSSP. How to adapt GAs to the JSSPs is very challenging but frustrating. Many efforts have been made in order to give an efficient implementation of GAs to the problem. In [(30)], a new GA is presented to solve the JSSP, while in [(4)] the impact of random initialization on solving the JSSP is addressed and using GA as an optimization technique.

Due to the NP-hard nature of the JSSP, using simple GA to solve the difficult problem may not be more efficient in practice. Much effort in the literature has focused on hybrid methods [(31)-(36)]. Ren and Yupingin [(31)] design some GA operators (mixed selection operator, new crossover operator and mutation operator based on the critical path) and solve JSSP more effectively. In [(32)] Athanasios and Stavros proposed a new hybrid parallel GA with specialized crossover and mutation operators utilizing path-relinking concepts from combinatorial optimization approaches. Some researchers focused their attention on combining the GA with local search schemes to develop some hybrid optimization strategies for JSSP [(35), (36)].

In this paper, we propose an effective hybrid algorithm for JSSP based on GA and local search. Firstly, we design a GA model for JSSP. We propose new initialization technique, modified crossover and mutation operators to improve the solution quality. Then local search based on the neighborhood structure is applied in the GA result[(7),(35)]. The proposed approach is tested by a set of standard instances taken from the literature drawn from the OR-library [(37), (38)]. The remainder of the paper is organized as follows. Detailed description of JSSP is presented in Section 2. In sections 3, GA is briefly introduced. In Section 4, the proposed algorithm is presented. Section 5 discusses the numerical results. Finally, we summarize the paper and present our future work in Section 6.

2. JOB SHOP SCHEDULING PROBLEM: The JSSP can be formulated as follows: For "n" jobs, each one is composed of several operations that must be executed on "m" machines. For each operation uses only one machine for a fixed duration. Each machine can process at most one operation at a time and once an operation starts processing on a given machine, it must complete processing on that same machine without any interruption. The operations of a given job have to be processed in a given set of order [(4)]. The problem's main aim is to schedule the activities or operations of the jobs on the machines such that the total time to finish all jobs, that is the makespan(Cmax), is minimized. The term makespan refers to the cumulative time to complete all the operations of all jobs on all machines. It is a measure of the time period from the starting time of the first operation to the ending time of the last operation. Sometimes there may be multiple solutions that have the minimum makespan, but the goal is to find out any one of them: it is not necessary to find all possible optimum solutions.

Let $J = 0,...,n,n+1$ denote the set of operations to be scheduled and $M = 1,...,m$ the set of machines. The operations 0 and $n+1$ are dummy, have no duration and represent the initial and final operations. The operations are interrelated by two kinds of constraints. First, the precedence constraints, which force each operation to be scheduled after all predecessor operations, P_j are completed. Second, operation can only be scheduled if the machine it requires is idle. Further, let d_j denote the (fixed) duration (processing time) of operation *j*. Let F_j represent the finish time of operation *j*. A schedule can be represented by a vector of finish time $(F_1, F_2, ..., F_{n+1})$. Let A(t) be the set of operations being processed at time *t*, and let if $r_{j,m} = 1$ operation *j* requires machine m to be processed and $r_{j,m} = 0$ otherwise [(7)].The conceptual model of the JSSP can be described the following way:

$$\text{Minimize } F_{n+1}(C_{\max}) \qquad (1)$$

$$\text{subject to :}$$

$$F_k \le F_j - d_j \qquad j = 1,...,n+1; \quad k \in P_j \qquad (2)$$

$$\sum_{j \in A\ t} r_{j,m} \le 1 \qquad m \in M; \qquad t \ge 0 \qquad (3)$$

$$F_j \ge 0, \ j = 1,...,n+1 \qquad (4)$$

The objective function (1) minimizes the finish time of operation *n+1* (the last operation), and therefore minimizes the makespan. Constraints (2) impose the precedence relations between operations and constraints (3) state that one machine can only process one operation at a time. Finally, (4) forces the finish times to be nonnegative.

The JSSP can be formally described by a disjunctive graph $G = V; C \cup D$, where *V* is a set of nodes representing operations of the jobs together with two special nodes, a source (0) and a sink (*), representing the beginning and end of the schedule, respectively. *C* is a set of conjunctive arcs representing technological sequences of the operations. *D* is a set of disjunctive arcs representing pairs of operations that must be performed on the same machines. As example, Tables I, II present the data for JSSP with three jobs and three machines. Table I includes the routing of each job through each machine while Table II includes the processing time for each operation. Figure I: presents the disjunctive graph of this example [(39), (40)].

Jobs	Machines		
J1	M1	M2	M3
J2	M1	M3	M2
J3	M2	M1	M3

Table I: Machine Sequence

Jobs	Time		
J1	3	3	3
J2	2	3	4
J3	3	2	1

Table II: Processing Time

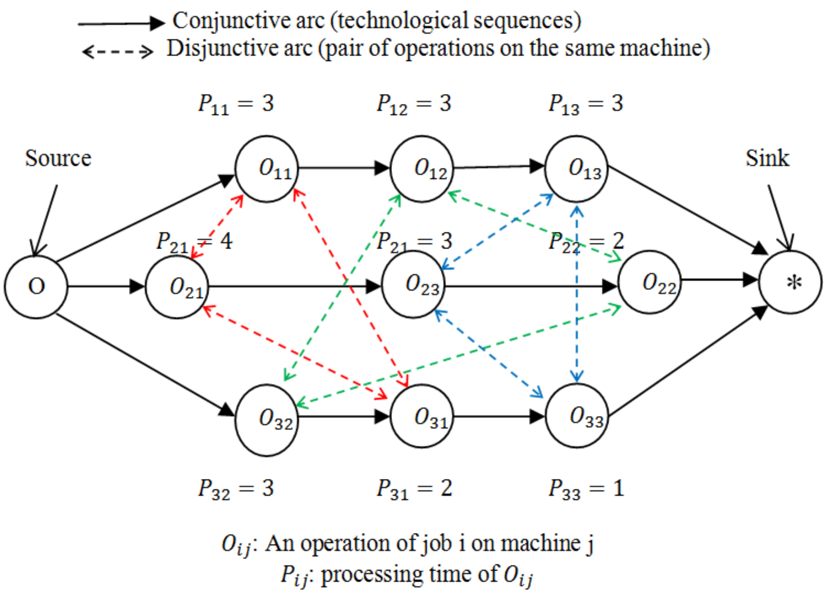


Fig I: The disjunctive graph of a 3 x 3 example

3. GENETIC ALGORITHM: GA is a powerful search technique based on natural biological evolution which is used for finding an optimal or near-optimal solution. The idea of GA was first proposed by Holland [(41)] in the early 1970s and since then has been widely used in solving optimization problem. GA has been implemented successfully in many scheduling problems. GA starts with a set of potential solutions (chromosomes). Next, genetic search operators such as selection, mutation and crossover are then applied one after another to obtain a new generation of chromosomes in which the expected quality over all the chromosomes is better than that of the initial generation [(42)]. This process is repeated until the termination criterion is met, and the best chromosome of the last generation is reported as the final solution. Then the genetic search steps can be described as shown in figure II [(43)]:

```
Generate an initial population
Check chromosome in constrains and repair out constrain values
Evaluate chromosome in the objective function
Do:
    New population [Select parents from population and recombine parents (Crossover and mutation operators)]
    Evaluate New population in the objective function
    Construct new generation from best of old population and New population
While satisfactory solution has been found
```

Fig II: The pseudo code of the general GA

4. THE PROPOSED ALGORITHM (HGA): The proposed algorithm is a combination between GA and local search. A new way of initializing the population of solutions is designed. A modified crossover and mutation operators are used to improve the solution quality. Then local search based on the neighborhood structure proposed by Nowicki and Smutnicki is applied in GA result [(44)]. Finally, the approach is tested on a set of standard instances taken from the literature. The computation results have validated the effectiveness of the proposed algorithm. The basic idea of the proposed algorithm can be described as follows:

4.1. GENETIC ALGORITHM

4.1.1. INTIAL POPULATION: For JSSP, the initial solution plays a critical role in determining the quality of final solution. However, the initial population has been produced randomly, it is not efficient. It will require longer search time to obtain an optimal solution and also decreases the search possibility for an optimal solution giving infeasible solutions [(45)]. In this paper, a new way of initializing first population of solutions is designed. This new method gives feasible solutions and is very useful in large problem. We generate the initial population (job sequence in machines) by depending on machine sequence for jobs. Arrange the jobs in machines that have earlier sequence in this machine. For example, problem in table I, for machine1 there are two jobs (1, 2) should start in it according to their machine sequence. In our approach we take this in account. Make the start in machine1with one of these jobs. Then complete jobs sequence in machine 1 with the same manner. Figure III illustrates initialization method with disjunctive graph. As shown in figure III, we order jobs in machine with the nearest jobs from start point (O). For example, the nearest jobs on machine 1 from start point (O) are job 1 and job 2 (yellow circles). So the job sequence in machine 1 will be (1,2,3),or (2,1,3). The same manner is used for the other machines.

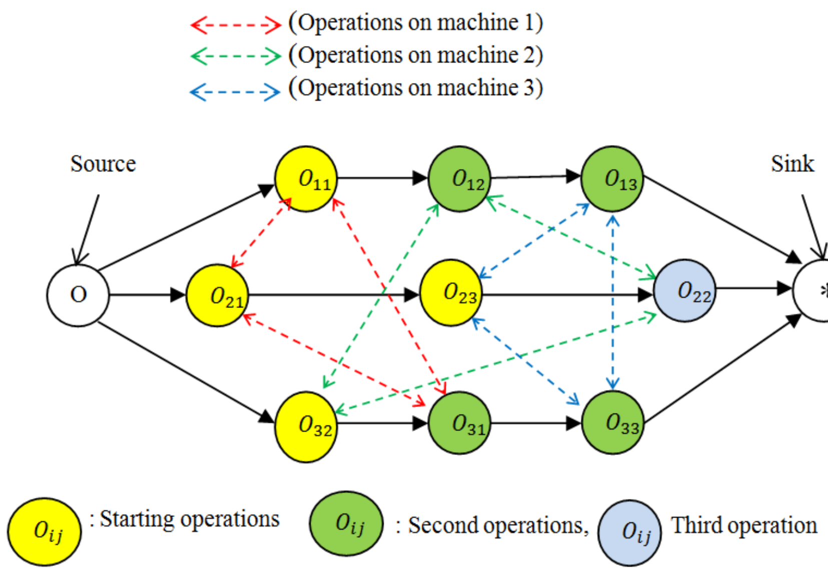

(Operations on machine 1)
(Operations on machine 2)
(Operations on machine 3)
Source
Sink
O
*
O_{11}
O_{12}
O_{13}
O_{21}
O_{23}
O_{22}
O_{32}
O_{31}
O_{33}
O_{ij} : Starting operations
O_{ij} : Second operations,
O_{ij} Third operation

Fig III: Initialization technique with disjunctive graph for the 3 x 3example

4.1.2. EVALUATION: In JSSP, makespan represents a good performance measure. Makespan of the jobs (Cmax) is the completion time of all the jobs where, the schedule with the minimal makespan often implies a high utilization of machines. The makespan objective is selected for comparison and assessing the performance of the generated solutions by GA.

4.1.3.CREAT NEW POPULATION: Creating a new population from the current population by ranking the solutions according to fitness function(makespan). Then we applied the three operators (selection, crossover, and mutation).

4.1.3.1. RANKING: Ranking individuals according to their fitness value, and returns a column vector containing the corresponding individual fitness value, in order to establish later the probabilities of survival that are necessary for the selection process.

4.1.3.2. SELECTION: For JSSP as any optimization problem there are several techniques of selection. The commonly used techniques for selection of individuals are roulette wheel selection, rank selection, steady state selection, stochastic universal sampling, etc. Here we used roulette wheel selection[(46)].

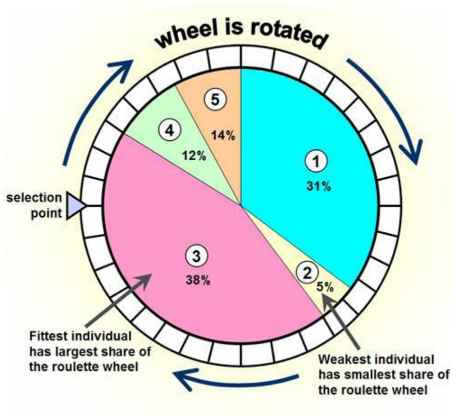

wheel is rotated
selection point
1
31%
2
5%
3
38%
4
12%
5
14%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

Fig IV: Roulette wheel selection method

In this method the parents are selected according to their fitness. Better chromosomes, are having more chances to be selected as parents. Each individual is assigned a slice of circular Roulette wheel, and the size of slice is proportional to the individual fitness of chromosomes, that is, bigger the value, larger the size of slice is. The Roulette wheel algorithm is started by finding the sum of all chromosomes fitness in the population. Then generate random number from the given population interval. Finally, go through the entire population and sum the fitness. When this sum is more than a fitness criteria value, stop and return this chromosome. Figure IV shows Roulette wheel for six individuals having different fitness values. The sixth individual has a higher fitness than any other, it is expected that the Roulette wheel selection will choose the sixth individual more than any other individual [(42)].

4.1.3.3. CROSSOVER: The crossover operates on two chromosomes at a time and generates offspring by combining features of both chromosomes. In general, the performance of the GAs depends, to a great extent, on the performance of the crossover operator used. During the past two decades, various crossover operators have been proposed for literal permutation encodings for JSSP, such as partial-mapped crossover (PMX), order crossover (OX), cycle crossover (CX), position-based crossover, order-based crossover, etc.[(47)]. In this paper, we use a modified OX that proposed by Gao and et al. [(48)], where they used OX in operation sequence; here we used it in job sequence for every machine. The modified OX can be described as follows:

Step 1: Select a subsection of job sequence for a machine from one parent at random.

Step 2: Produce offspring by copying the substring of job sequence for a machine into the corresponding positions.

Step 3: Delete the jobs that are already in the substring from the second parent. The resulted sequence of jobs contains jobs that the offspring needs.

Step 4: Place the jobs into the unfixed positions of the offspring from left to right according to the order of the sequence in the second parent. The modified OX for one machine is illustrated in Figure V.

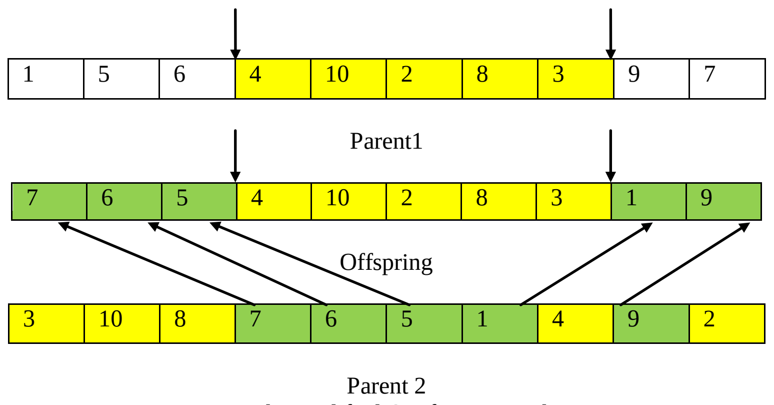


Fig V: The modified OX for one machine.

4.1.3.4. MUTATION: Mutation is just used to produce small perturbations on chromosomes in order to maintain the diversity of population. During the last decade, there are several mutation operators have been proposed such as inversion, insertion, displacement, reciprocal exchange mutation, and shift mutation [(48)]. Inversion mutation selects two positions within a chromosome at random and then inverts the substring between these two positions. In this paper we advanced inversion mutation, by choosing two or more positions in the jobs sequence for one machine and invert them as showed in Figure VI. These steps are applied to other machines, to get a new Offspring.

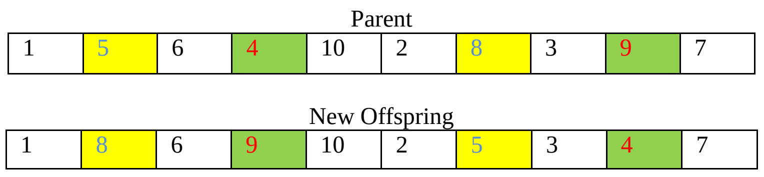


Fig VI: Inversion mutation on job sequence for one machine.

4.1.4. NEW GENERATION: In this step, the new generation composed by migration initial population with and offspring population. Then we take the best individuals of initial population with generation gap percentage and offspring population.

4.1.5. TERMINATION TEST: The algorithm is terminated either the maximum number of generations is achieved, or when the individuals of the population converge, convergence occurs when all individual positions in the population are identical. In this case, crossover will have no further effect. Otherwise, return to step 4.3.

4.2. LOCAL SEARCH PROCEDURE: Using GAs for the JSSP are usually with a slow convergence speed and easy to trap into local optimal solutions. In order to enhance the convergence speed, we combine the GA with local search schemes to develop some hybrid optimization strategies for JSSP improving the solution quality. We use the approach of Nowicki and Smutnicki [(37), (38)].The local search procedure starts by finding the critical path for the solution obtained by GA. Critical path can be determined in the Gantt-Chart representation for the solution. The critical path decomposes by a number of blocks where a block is a maximal sequence of adjacent operations that require the same machine. Any operation on the critical path is called a critical operation. Secondly, a neighborhood is defined as interchanges of any two sequential operations. Interchanging the last two or the first two critical operations gives better solution. The local search procedure is shown in Figure VII, while Figure VIII shows the flow chart of the proposed algorithm (HGA).

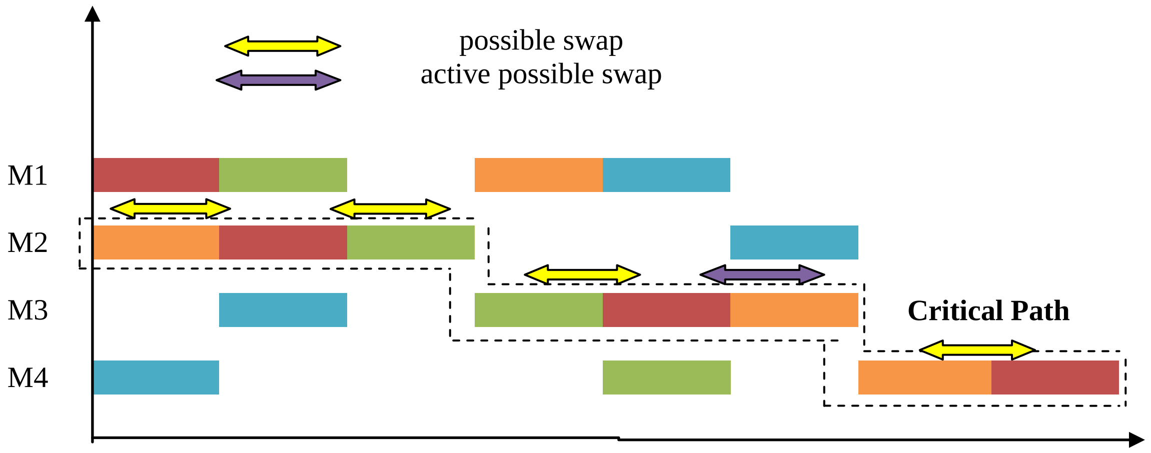
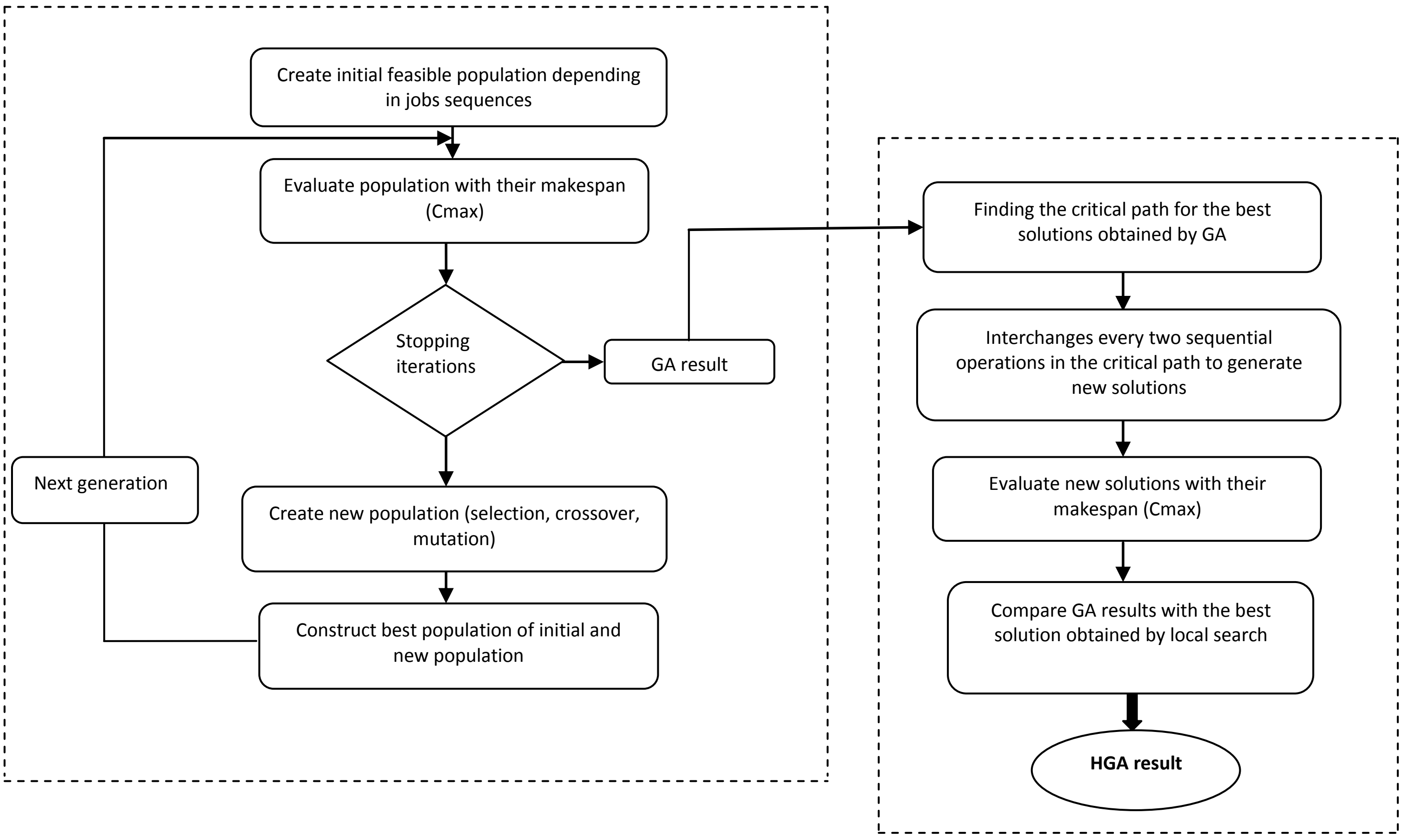


Fig VII: the local search procedure


Create initial feasible population depending in jobs sequences
Evaluate population with their makespan (Cmax)
Stopping iterations
GA result
Next generation
Create new population (selection, crossover, mutation)
Construct best population of initial and new population
Finding the critical path for the best solutions obtained by GA
Interchanges every two sequential operations in the critical path to generate new solutions
Evaluate new solutions with their makespan (Cmax)
Compare GA results with the best solution obtained by local search
HGA result

FigVIII:The flow chart of the proposed algorithm (HGA)

5. NUMERICAL STUDIES: To illustrate the effectiveness of the proposed algorithm, it is tested on a set of standard instances (JSSP test problems) taken from the literature [(37), (38)].As any non-traditional optimization algorithms, the proposed algorithm, involves a number of parameters that affect the performance of algorithm. The parameters are summarized in Table III. The algorithm is coded in MATLAB 7.8 and the simulations have been executed on an Intel core (TM) i7-4510u cpu, 2.00GHZ -2.60 GHz processor.

Generation gap	0.9
Crossover rate	0.9
Mutation rate	0.7
Selection operator	roulette wheel selection Single point
Crossover operator	order crossover
Mutation operator	Inversion mutation
GA generation	50-1000

Table III: Summary of the parameters used in the proposed algorithm

Table IV shows the problem name and size(number of jobs ×number of operations), the best known solution (BKS), results obtained by applying GA only, the results obtained by HGA and the improvement percentage in the obtained results due to hybrid local search with GA. In addition, Figure IX showed the improvement percentage for each problem. The Gantt charts of some obtained results by our approach for the problems (FT06-LA01-LA05) are illustrated in Figure X-XII. From the table and figures, we can say that hybridizing the local search with GA improves the solutions quality; where the mean improvement percentage is about 5.36%. Furthermore, we can see that the solutions obtained by HGA are better and converge to the BKS than solutions obtained by GA only.

Problem (size)	Best Known Solution (BKS)	Results of GA	Results of HGA	Improvement Percent.
Ft06 (6×6)	55	55	55	0.000%
LA01 (10×5)	666	666	666	0.000%
LA02 (10×5)	655	790	714	11.603%
LA03 (10×5)	597	683	617	11.055%
LA04 (10×5)	590	672	632	6.779%
LA05 (10×5)	593	593	593	0.000%
LA06 (15×5)	926	926	926	0.000%
LA07 (15×5)	890	916	899	1.91%
LA08 (15×5)	863	863	863	0.000%
LA09 (15×5)	951	951	951	0.000%
LA10 (15×5)	958	958	958	0.000%
LA11 (20×5)	1222	1222	1222	0.000%
LA12 (20×5)	1039	1039	1039	0.000%
LA13 (20×5)	1150	1150	1150	0.000%
LA14 (20×5)	1292	1292	1292	0.000%
LA15 (20×5)	1207	1324	1311	1.077%
LA16 (10×10)	945	1140	1077	6.667%
LA17 (10×10)	784	848	843	0.638%
LA18 (10×10)	848	948	894	6.368%
LA19 (10×10)	842	961	896	7.719%
LA20 (10×10)	902	956	949	0.776%

Table IV: Comparison between BKS, and the proposed algorithm results (GA only and HGA) with Improvement Percent.

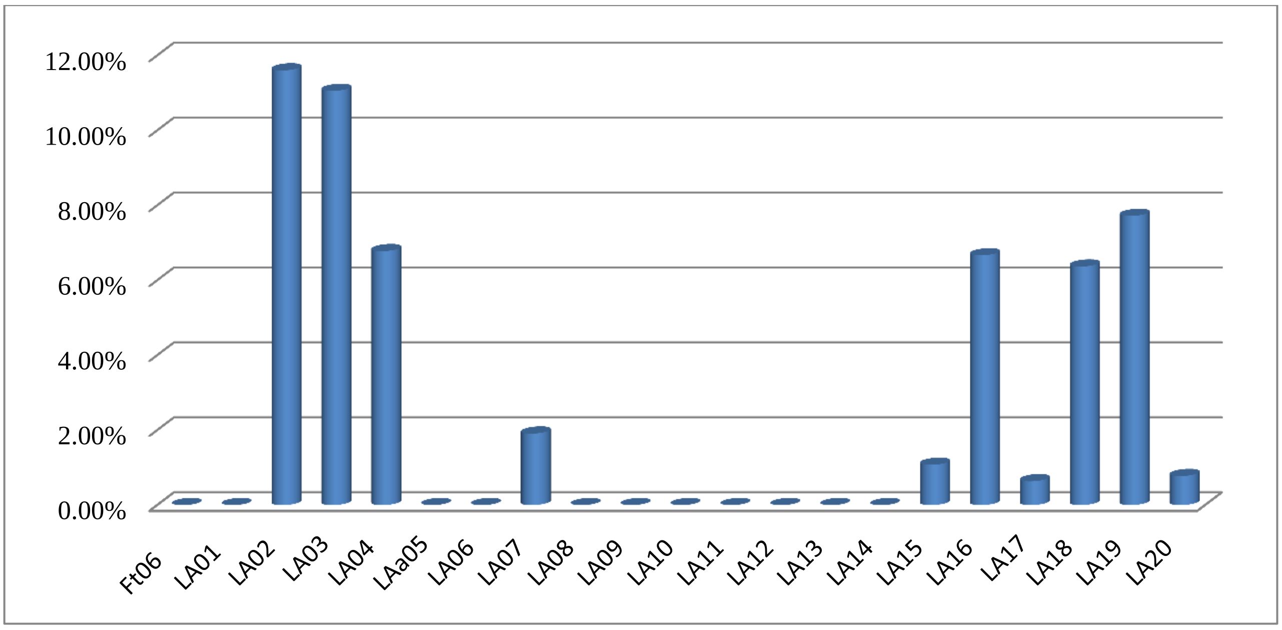


Fig IX: The improvement percentage due to hybrid of local search with GA.

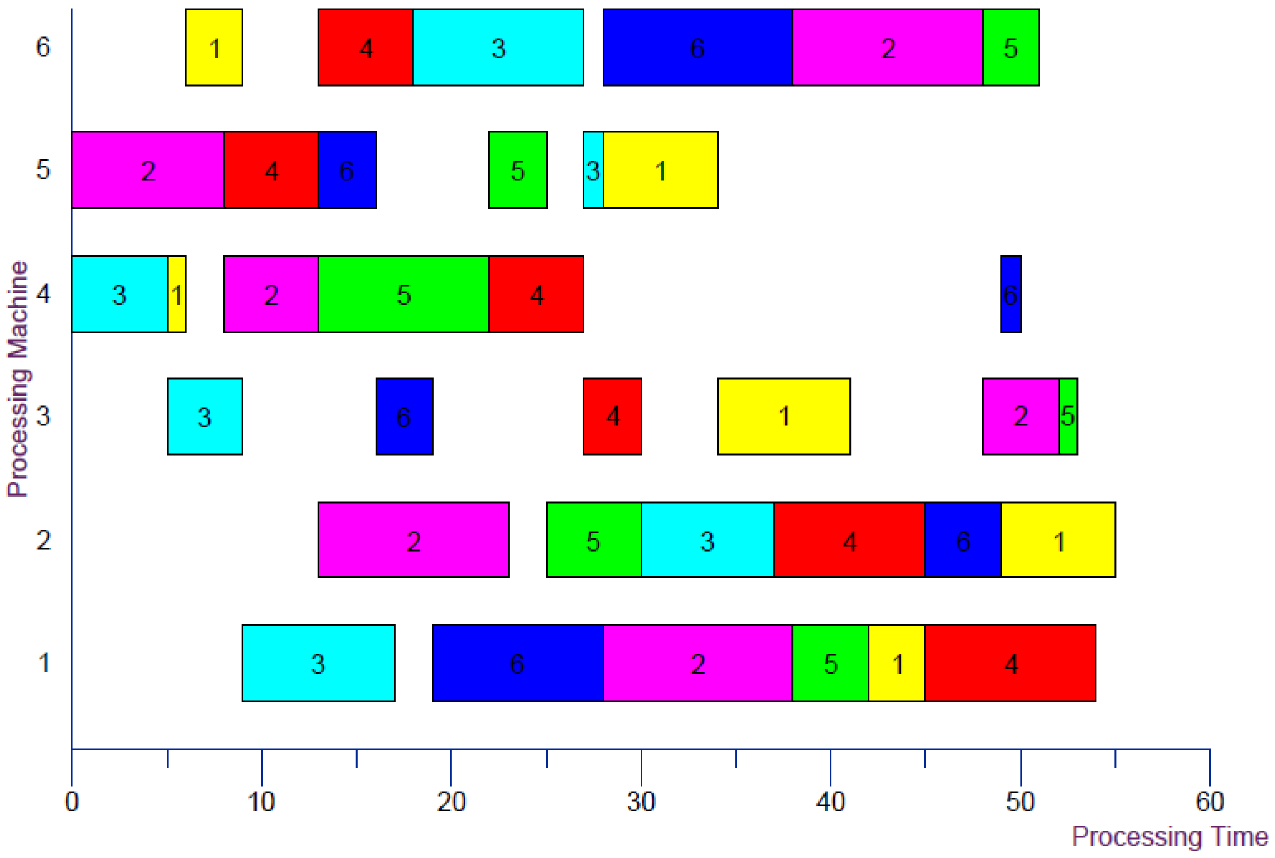


Fig X: Gantt chart of the obtained solution for the problem FT06.

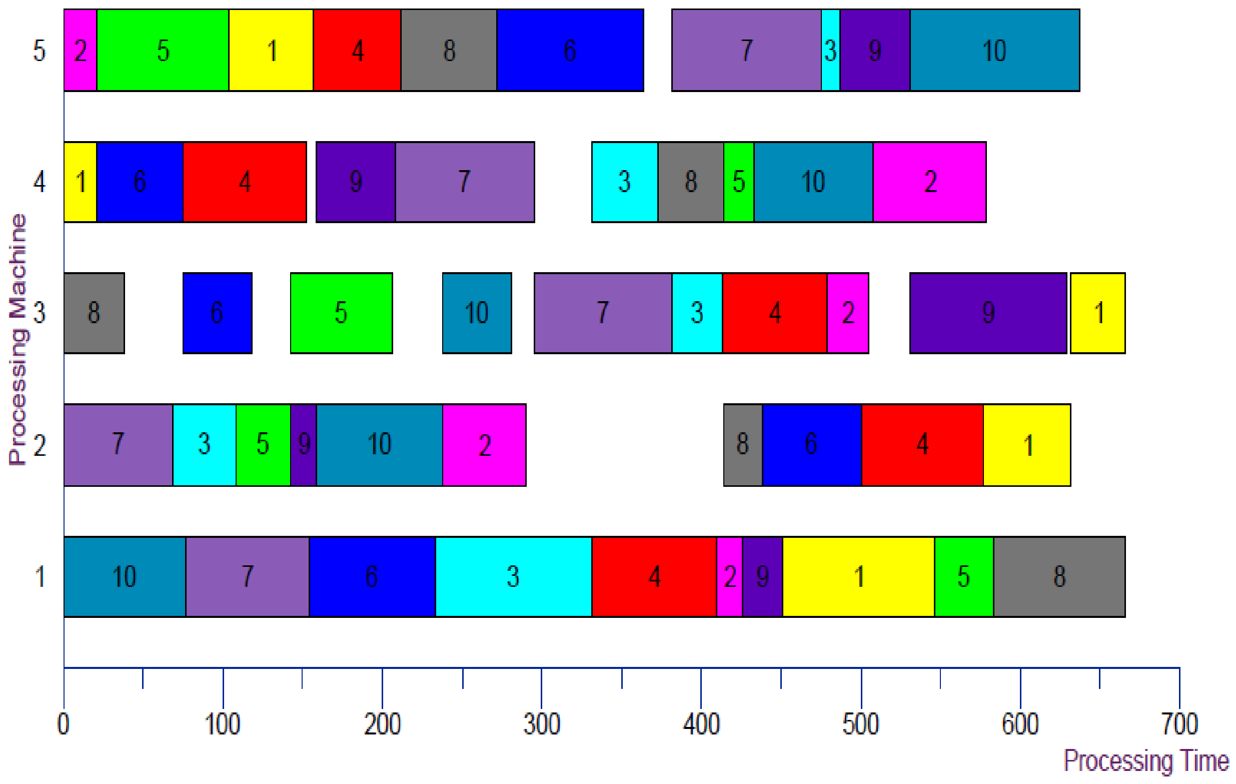


Fig XI: Gantt chart of the obtained solution for the problem LA01.

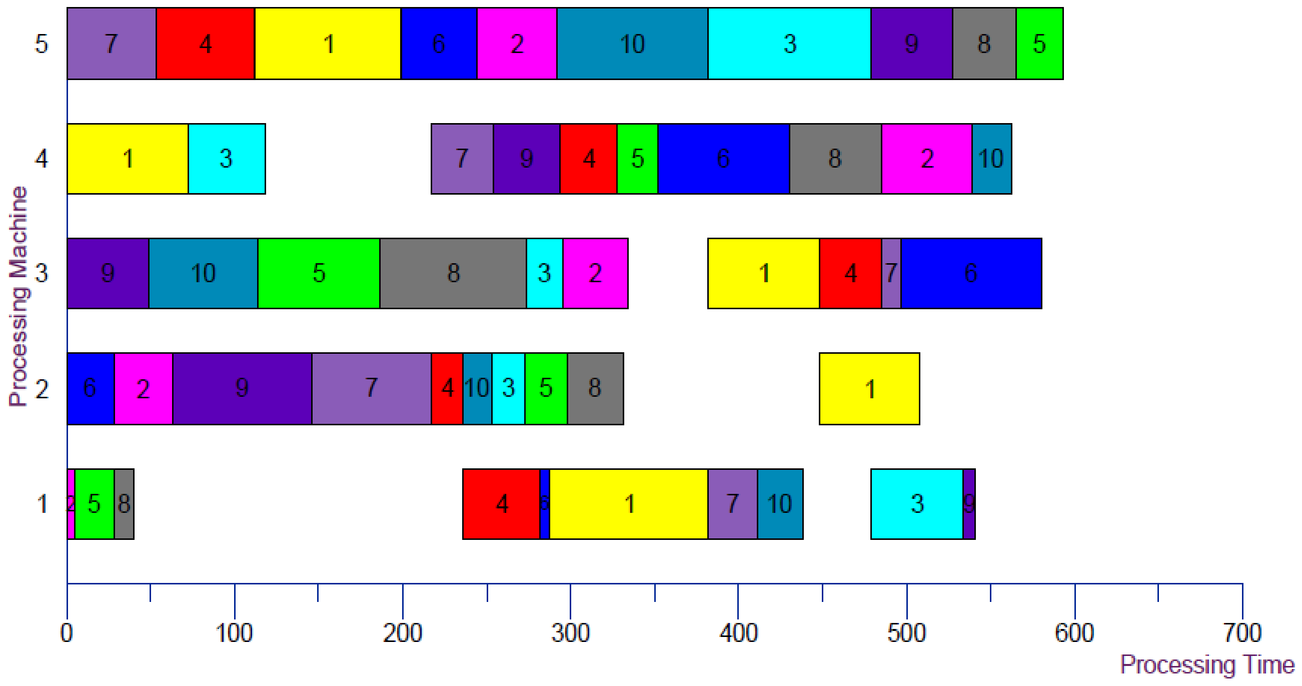


Figure XII: Gantt chart of the obtained solution for the problem LA05

The non-deterministic nature of our algorithm makes it necessary to carry out multiple runs on the same problem instance in order to obtain meaningful results. We ran our algorithm 15times for each test problem. Worst, best, mean and standard deviation (Std.) of results are illustrated in Table V. The standard deviation is a very important problem in practical production area. In order to ensure our scheduling plan performs effectively, the standard deviation should be as small as possible. Through simulation, we find the standard deviation is small which indicates that HGA possesses excellent robustness.

Problem	Worst	Best	Mean	Std.
Ft06	55	55	55	0
LA01	680	666	674.3000	5.0210
LA02	758	714	729	18.7457
LA03	671	617	653.1000	11.5884
LA04	672	632	630.7000	3.9000
LA05	593	593	593	0
LA06	593	926	593	0
LA07	973	899	934.5714	22.7564
LA08	863	863	863	0
LA09	951	951	951	0
LA10	958	958	958	0
LA11	1222	1222	1222	0
LA12	1039	1039	1039	0
LA13	1150	1150	1150	0
LA14	1292	1292	1292	0
LA15	1379	1311	1349.4	22.9371
LA16	1107	1077	1088.6	11.2427
LA17	877	843	851.929	3.9362
LA18	954	894	930.0714	12.8072
LA19	1015	896	961	42.0077
LA20	1056	949	991	18.3011

Table V: Mean, worst, best and standard deviation for the test instances.

For comparison purposes, Table VI shows the results and the percentage relative error with respect to BKS obtained by proposed algorithm as compared to CGS [(28)]. In addition, Figure XIII proposed the percentage relative error for the two algorithms CGS and HGA. It is quite evident that our algorithm more converges to the BKS and gives comparable minimum relative error or better than those obtained by CGS; where HGA reached to the BKS for 52.4% of its solved test problems while the CGS reached to the BKS for 26.3% and in the major remained problems HGA reached near to BKS than CGS.

PROBLEM	CGS	HGA	Percentage relative error(CGS)	Percentage relative error(CGA)
FT06	-	55	-	0.000%
LA01	713	666	7.057%	0.000%
LA02	757	714	15.572%	9.008%
LA03	682	617	14.238%	3.350%
LA04	669	632	13.389%	7.119%
LA05	593	593	0.000%	0.000%
LA06	926	926	0.000%	0.000%
LA07	-	877	-	1.685%
LA08	897	863	3.939%	0.000%
LA09	956	951	0.526%	0.000%
LA10	958	958	0.000%	0.000%
LA11	1222	1222	0.000%	0.000%
LA12	1058	1039	1.829%	0.000%
LA13	1152	1150	0.174%	0.000%
LA14	1292	1292	0.000%	0.000%
LA15	1310	1311	8.536%	8.616%
LA16	1010	1077	6.878%	13.968%
LA17	-	843	-	7.526%
LA18	914	894	7.783%	5.425%
LA19	944	896	12.114%	6.413%
LA20	949	972	5.211%	7.761%

Table VI:Results and the percentage relative error for the two algorithms CGS and HGA

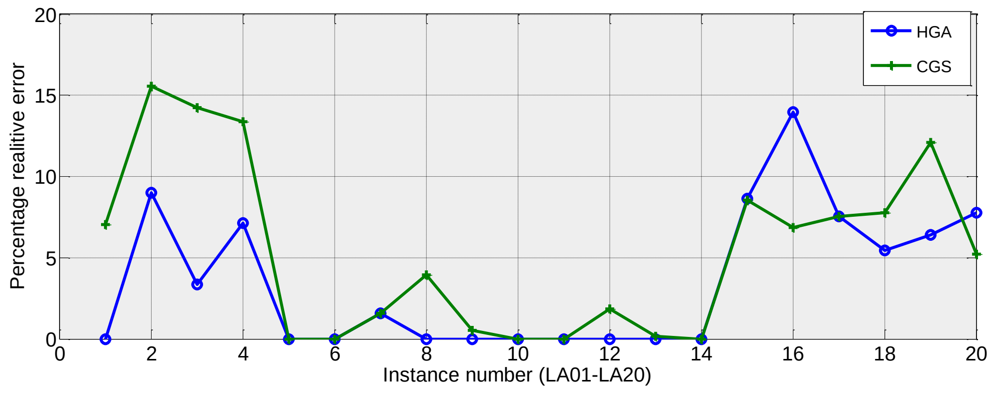


Fig XIII: The comparison between the percentage relative error for the two algorithms CGS and HGA

Overall, we applied the proposed algorithm in various test instances with different size. In general, the results have shown the correctness, feasibility and the usability of the proposed algorithm. Noting also that increasing the problem size doesn't guarantee that the proposed algorithm reach to the BKS due to using 2.00GHZ 2.60 GHz processor. But, HGA integrates the powerful global searching of GA with the powerful local searching of local search. In addition, Integrating GA with local search accelerates the optimum seeking operation and speeds the convergence to the BKS. Finally, due to simplicity of HGA procedures, it can be used to handle engineering applications.

6. CONCLUSION: In this paper we present a combination between GA and local search for solving the Job-shop scheduling problem. Firstly, a new initialization method is proposed. A modified crossover and mutation operators are used. Secondly, local search based on the neighborhood structure is applied in the GA result. Finally, the approach is tested on a set of standard instances taken from the literature. The results obtained by HGA are better and converge to the BKS than solutions obtained by GA only, which mean that hybridizing local search with GA improves the solutions quality. In addition, the proposed algorithm is more converges to the BKS and gives comparable minimum relative error or better than those obtained by other approach. In general, the results have shown the correctness, feasibility and the usability of the proposed algorithm.
In our future works, the following will be researched:
a) Solving larger scale examples to demonstrate the efficiency of the proposed algorithm.
b) Hybridizing more optimization techniques with the proposed algorithm to accelerate the convergence property and improve the solution quality.
c) Updating the proposed algorithm to solve the JSSP as multi-objective optimization problem.

ACKNOWLEDGEMENT: We would like to express or sincere gratitude to the anonymous reviewers for their efforts and comments, which helped us considerably in improving the quality of this paper.

REFERENCES:

1. Kolharkar S., ZanwarD.R., 2013, *Scheduling in Job Shop Process Industry,* Journal of Mechanical and Civil Engineering, 5, 2278-1684.
2. PinedoM.L., 2009, *Planning and scheduling in manufacturing and services*, Springer, second edition, 207-339.
3. Chryssolouris G., Subramaniam V., 2001, *Dynamic scheduling of manufacturing job shops using genetic* algorithms, Journal of Intelligent Manufacturing , 12,281-293.
4. Tamilarasi A., Jayasankari S., 2012*, Evaluation on GA based Model for solving JSSP*, International Journal of Computer Applications, 43(7), 975 – 8887.
5. Garey, M.R., Johnson, D.S. and Sethi R., 1976, The Complexity of Flow-shop and Job-shop Scheduling, Mathematics of Operations Research,1, 117-129.
6. Garey M.R., Johnson D.S., 1979, *Computers and intractability: A guide to the theory of NP-completeness*, first edition, W. H. Freeman & Co. New York, NY, USA, 359-376.
7. Goncalves J.F., Mendes J.J.M., Resende M.G.C., 2005, *A hybrid genetic algorithm for the job shops scheduling problem*, European Journal of Operational Research, 167, 77–95.
8. El-Shorbagy M.A., Mousa A.A., Nasr S.M., 2016, *A Chaos-Based Evolutionary Algorithm for General Nonlinear Programming Problems,* Chaos, Solitons and Fractals, 85, 8–21.
9. Farag M.A., El-Shorbagy M.A., El-Desoky I.M., El-Sawy A.A. and Mousa A.A.,2015, *Binary-Real Coded Genetic Algorithm Based K-Means Clustering for Unit Commitment Problem*, Applied Mathematics, 6, 1873-1890.
10. Mousa A.A., El_Desoky I.M., 2013, *Stability of Pareto Optimal Allocation of Land Reclamation by Multistage Decision-Based Multipheromone Ant Colony Optimization*, Swarm and Evolutionary Computation, 13, 13–21.
11. Mousa A.A., El-Shorbagy M.A., Abd El-WahedW.F., 2012, *Local Search Based Hybrid Particle Swarm Optimization for Multiobjective Optimization*, International journal of Swarm and evolutionary computation, 3, 1-14.

12. Mousa A.A., El-Shorbagy M.A., 2012, *Enhanced Particle Swarm Optimization Based Local Search for Reactive Power Compensation Problem*, Applied Mathematics 3, 1276-1284.
13. El-Sawy A.A., Hendawy Z.M., El-Shorbagy M.A.,2012, *Trust-Region Algorithm Based Local Search for Multi-Objective Optimization*. Proceedings of the first International Conference on Innovative Engineering Systems (ICIES), Egypt, 207-212.
14. Abd El-Wahed W.F., Mousa A.A., El-Shorbagy M.A., 2011, *IntegratingParticle Swarm Optimization with Genetic Algorithms for Solving Nonlinear Optimization Problems*, Journal of Computational and Applied Mathematics, 235, 1446–1453.
15. El-Shorbagy M.A., Mousa A.A., Abd-El-Wahed W.F., 2011, *Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Optimization*, Lambert academic publishing GmbH &Co.kG, Berlin.
16. Osman M.S., Abo-Sinna M.A., Mousa A.A., 2006, *IT-CEMOP: An Iterative Co-Evolutionary Algorithm for Multiobjective Optimization Problem with Nonlinear Constraints*, Journal of Applied Mathematics & Computation (AMC), 183, 373-389.
17. Zhang C., Rao Y., Li P., 2008, *An effective hybrid genetic algorithm for the job shop scheduling problem,* The International Journal of Advanced Manufacturing Technology, 39, 965–974.
18. Yan-Fang Y., Yue Y., 2015, *An improved genetic algorithm to the job shop scheduling problem*, Journal of Chemical and Pharmaceutical Research, 7(4), 322-325.
19. Song S.-Z., Ren J.-J., Fan J.-X., 2012, *Improved simulated annealing algorithm used for job shop scheduling problems,* in Advances in Electrical Engineering and Automation, Springer, 17–25.
20. Thamilselvan R., Balasubramanie P*., 2012, Integrating Genetic Algorithm, Tabu Search and Simulated Annealing for Job ShopScheduling Problem*, International Journal of Computer Applications, 48(5), 975 – 888.
21. Mehmood N., Umer M., Ahmad R., Rafique A.F., 2013,*Optimization of Makespan and Mean Flow Time for Job Shop Scheduling Problem FT06 Using ACO*, Life Science Journal,10(4), 477-484.
22. Nazif H., 2015, *Solving job shop scheduling problem using an ant colony algorithm*, Journal of Asian Scientific Research, 5(5), 261-268.
23. FlórezE., Gómez W.,Bautista L., 2013,*Anant colony optimization algorithm for job shop scheduling problem*, International Journal of Artificial Intelligence & Applications,4(4), 53-66.
24. Jaziri W., 2008, Some *New Results on Tabu Search Algorithm Applied to the Job-Shop Scheduling Problem*, Tabu Search, I-Tech Education and Publishing.
25. Zhang C., Li P., Guan Z., Rao Y.,2007, *A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem*, Computers & Operations Research, 34(11), 3229–3242.
26. Sha D.Y., Hsu C.-Y., 2006,*A hybrid particle swarm optimization for job shop scheduling problem*, Computers & Industrial Engineering, 51,4791–808.
27. Zhang G., Shao X., Li P.,Gao L., 2009,*An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem*, Computers & Industrial Engineering, 56, 1309–1318.
28. Deepanandhini D., Amudha T.,2013,*Solving job-shop scheduling problem with consultant guided search metaheuristics,* International Journal of Software and Web Sciences, 3 (1), 1-6.
29. Lazár I., 2012, *review on solving the job shop scheduling problem techniques: recent development and trends*, Transfer inovácií, 23, 55-60.
30. LI Y., CHEN Y., 2010, *A Genetic Algorithm for Job-Shop Scheduling*, journal of software, 5, 269-247.
31. Qing-dao-er-jiR., Wang Y., 2012*, A new hybrid genetic algorithm for job shop scheduling problem*, Computers & Operations Research 39, 2291–2299.
32. Spanos A.C., Ponis S.T., TatsiopoulosI. P., Christoul.T., Rokou E., 2014, *A new hybrid parallel genetic algorithm for the job-shop scheduling problem*, International transactions in operational research, 21, 479–499.
33. Abu-Srhahn A., Al-Hasan M., 2015, *Hybrid Algorithm using Genetic Algorithm and Cuckoo Search Algorithm for Job Shop Scheduling Problem*, Internationl Journal of Computer Science Issues,12, 288-292.
34. Moin N.H., Sin O.C., Omar M.,2015, *Hybrid Genetic Algorithm with Multiparents Crossover for Job Shop Scheduling Problems*, Mathematical Problems in Engineering, Volume 2015.
35. OmbukiB.M., Ventresca M., 2004, Local Search Genetic Algorithms for the Job Shop Scheduling Problem, Applied Intelligence, 21, 99-109.
36. Camino R.V., Varela R. , González M.A., 2010, Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times, Journal of Heuristics,16, 139-165.
37. Wang X., Duan H., 2014, *A hybrid biogeography-based optimization algorithm for job shop scheduling problem*, Computers & Industrial Engineering, 73, 96–114.
38. Beasley E.J., 1990, *OR-library: distributing test problems by electronic mail, Journal of the Operational Research Society* 41 (11), http://mscmga.ms.ic.ac.uk.
39. Gen M., Cheng R., 1997, *Genetic algorithms and engineering design*, Wiley, 1-90.
40. Balas E., 1969, *Machine sequencing via disjunctive graphs: an implicit enumeration algorithm*, Operations Research, 17, 941–57.
41. Rao S.S., 2009, *Engineering Optimization: Theory and Practice,* Wiley, 3rd edition, 693-702.
42. Malhotra R., Singh N., Singh Y., 2011, *Genetic algorithms: Concepts, design for optimization of process controllers*, Computer and Information Science, 4(2), 39-54.
43. Nasr S.M., El-Shorbagy M.A., El-Desoky I.M., HendawyZ.M., Mousa A.A., 2015, *Hybrid genetic algorithm for constrained nonlinear optimization problems*, British journal of mathematics & computer science, 7(6), 466-480.
44. Starkweather T., Whitley D., Mathias K., McDaniel S., 1992, *Sequence scheduling with genetic algorithms*, Proceedings of the US/German Conference on New Directions for OR in Manufacturing, 130-148.
45. Park B. J., Choi H.R., Kim H.S., 2003, *A hybrid genetic algorithm for the job shop scheduling problem*, Computers & Industrial Engineering, 45, 597–613.
46. Baker J.E., 1987, *Reducing bias and inefficiency in the selection algorithm*, proceedings of the second international conference on genetic algorithms. Morgan Kaufmann Publishers, 14-21.
47. Cheng R., Gen M., Tsujimura Y., 1999, *A tutorial survey of job-shop scheduling problems using genetic algorithms*, Computers & Industrial Engineering, 36, 343-364.
48. Gao J., Sun L., Gen M., 2008, *A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems*, Computers & Operations Research, 35, 2892 – 2907.