Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 3, No. 4, November 2012), Pages: 17-24
www.jacr.iausari.ac.ir

# A Hybrid Genetic Algorithm for the Open Shop Scheduling with Makespan and Total Completion Time

**Behnam Barzegar[1], Homayun Motameni [2], Ali Khosrozadeh ghomi[3], Azadeh Divsalar[2]**

*1) Department of Computer Engineering, Nowshahr Branch, Islamic Azad University, Nowshahr, Iran*
*2) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran*
*3) Department of Computer Engineering, Science and Research Ayatollah Amoli Branch, Islamic Azad University, Amol, Iran*

barzegar@iauns.ac.ir ; motameni@iausari.ac.ir; alikhosrozadeh@gmail.com; azadeh.divsalar@gmail.com

**Received: 2012/07/24; Accepted: 2012/10/04**

**Abstract**

*Proper scheduling of tasks leads to optimum using of time and resources, in order to obtaining best result. One of the most important and complicated scheduling problems is open shop scheduling problem. There are n jobs in open shop scheduling problem which should be processed by m machines. Purpose of scheduling open shop problem is attaining to a suitable order of processing jobs by specified machines so that makespan can be minimized. Open shop scheduling problem has very large and complex solution space and so is one of NP-Problems. Till now, different algorithms have been presented for open shop scheduling problem. In this paper, we have used combined genetics algorithm as a strategy for solving scheduling open shop problem and compared proposed algorithm with DGA algorithm. Results show that the proposed algorithm has better effectiveness than DGA algorithm.*

***Keywords:*** *Open shop, Genetics Algorithm, Scheduling*

## 1. Introduction

One of the most important and complicated scheduling problems is scheduling open shop problem. Scheduling open shop problem has many application in engineering field, especially in industrial cases. There are n jobs in scheduling open shop problem which each job includes set of operations. Each operation should be processed by a specific machine in determined time. Scheduling open shop problem has very large and complicated solution scope and belongs to NP problems. Purpose of scheduling open shop problems is finding a reasonable combination of job process, so that required time for makespan could be minimized. Many researchers have presented different algorithms which we point out some of them as follows:

O(n) for two machines[2]. Pinedo offered another simple distribution rule known as "Longest Alternate Processing Time (LAPT) "[3] which solves problem for two machines in polynomial time. Brucker [4] extended other branch and connected algorithm to general problem of m machine. Fiala presented polynomial time algorithm which solves problem for m arbitrary machines [5]. Among innovative algorithms, Alcaide and his coworkers [6] presented a tab searching algorithm for minimizing makespan in scheduling open shop problem. Liaw [7] developed a connected genetics algorithm for solving scheduling open shop problem in order to minimize makespan. Prins [8] achieved a fine solution for solving scheduling open shop problem by providing a genetics algorithm.

Among above mentioned methods, genetics algorithm is considered as one of the best-known applied algorithms, but has got some disadvantages like instability and low speed in finding optimal solution.

In this paper, a new algorithm is presented as a combination of genetics algorithm with tab searching algorithm. This combination results in a better searching in problem solution space in order to achieve a better scheduling.

In the second section problem explanation has presented. Then, in the third section the proposed algorithm is discussed and in the fourth part, implementation results have been offered and finally conclusion ends the paper.

## 2. Problem explanation

Scheduling open shop problem includes n jobs and m machines and each job includes m operations which should be implemented in its corresponding machine in already determined time. In other words, first operation of job j should be implemented by first machine and second operation of job j should be implemented by second machine and so on. Purpose of scheduling open shop problem is to get a suitable combination of job processing in order for makespan to be minimized.

Table 1 shows typical system of scheduling open shop problem which includes three jobs and each job includes 3 operations which should be implemented.

***Table 1. A typical system of open shop***

| Jobs | Machine 1 | Machine 2 | Machine 3 |
|---|---|---|---|
| Job 1 | 7 | 12 | 16 |
| Job 2 | 13 | 10 | 13 |
| Job 3 | 18 | 14 | 2 |

For example, here, operation 2 form job 2 whose required running time is 9, should be implemented by machine 2. Figure 1 illustrates Gant diagram of table 1 typical system.

As illustrated in figure 1 Gant diagram, j 1 O 3 (operations of job 1) has been processed in time 0 by machine 2, as well as j 1 O 1 (operation 1 of job 1) has been processed in time 13 by machine 2 in 5 cases makespan in this problem is 12.

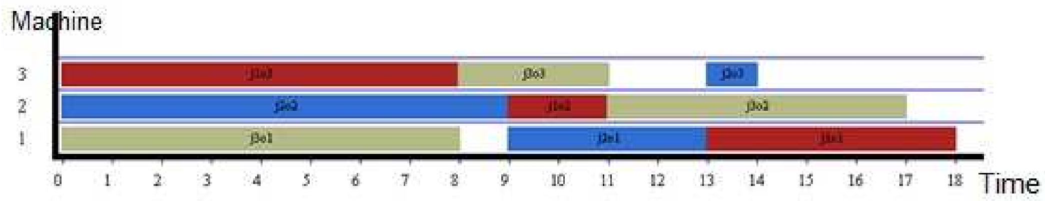

***Figure1. Gant Diagram, Sample system of Table 1***

## 3. Proposed Algorithm

In our proposed method we used of combination of genetics algorithm with local searching algorithm as a strategy for solving scheduling open shop problem which improves searching in state space, and its objective is minimizing makespan. Figure 2 shows the total stages of proposed algorithm.

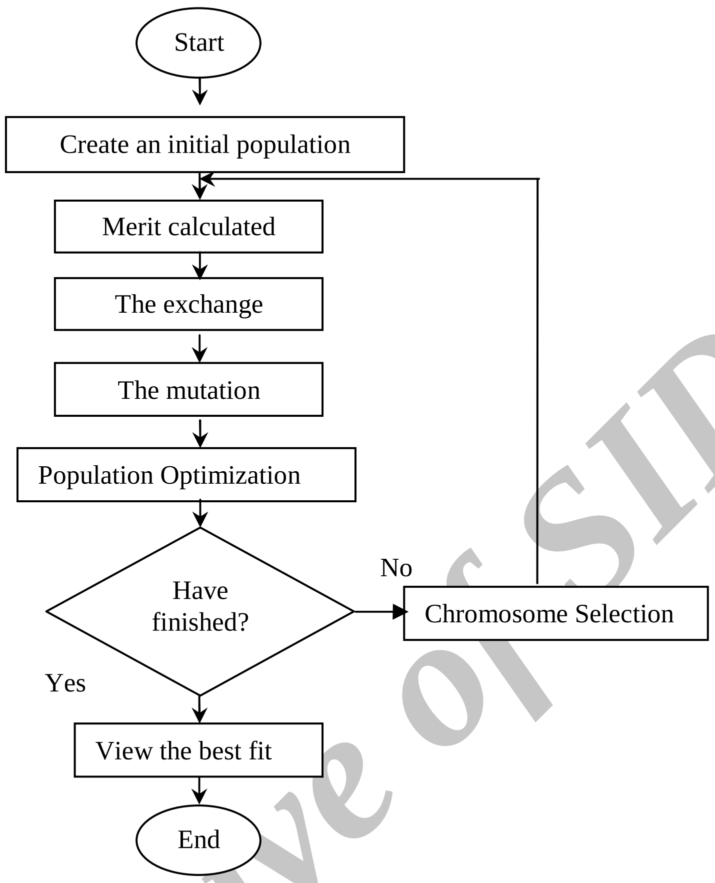

***Figure2. Overall the proposed algorithm***

### *3.1 Chromosome demonstration*

In the proposed algorithm, a one dimensional array in the length of makespan for chromosome is used. In this method, each chromosome is a unique integer number which is obtained by numbering job operations. Table 2 shows way of numbering operations for typical system of Table 1.

***Table 2. Number of system operations, Table 1***

| Job | Machine | Operation# |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 1 | 3 | 3 |
| 2 | 1 | 4 |
| 2 | 2 | 5 |
| 2 | 3 | 6 |
| 3 | 1 | 7 |
| 3 | 2 | 8 |
| 3 | 3 | 9 |

Figure 3 shows a typical chromosome for typical system of Table 1. As you could see in the structure of chromosome in figure 3, each gene presents operation number that should be processed by the corresponding machine.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | 3 | 5 | 4 | 9 | 2 | 6 | 1 | 8 |

***Figure3. Structure of typical chromosome***

For instance, First gene number is 1, which in fact points out operation 1 from job 3 that should be implemented by machine 1.

In this method, each chromosome shows one scheduling for makespan.

### *3.2 Fitness*

Chromosome fitness is considered as required time for makespan. In our proposed algorithm we used the following formulation for calculating fitness.

$$\text{Fitness} = \text{Max}_{1<i<=N} \{ T_i \} \qquad (3.1)$$

Where N is number of jobs and T is makespan.

### *3.3 Parents Selection*

In proposed algorithm, we used an ordering method to select parents. In this technique, all chromosomes are ordered based on fitness according to the following formulation.

$$(\text{Max} - \text{Fit}_i) + 1 \; \text{cr}_{1< =I <=n} \qquad (3.2)$$

Where $cr_i$ is order of chromosome I, Max is worst fitness, and $Fit_i$ is fitness of chromosome i.

The best chromosome gives Max-Fit+1 order and worst chromosome received order 1.

Therefore, all the chromosomes in this method have a chance of selecting.

### *3.4 Crossover-Over Operator :*

In proposed algorithm, after selecting 2 parents for crossover operator, we selected a random number within the 1 through n interval (n is number of operators) and those genes which are in the range of 0 to random number shift from primary parent, similar genes of second parent are eliminated, and the remained genes are inserted in empty houses. Figure 4 shows crossover operator for a typical system of Table 1. The advantage of using method is that it does not generate child with repeated genes, and also in each cross over just one child would be generated.

### *3.5 Mutation Operator*

After selecting parent chromosome from existing population, 2 randomized points in the interval 1 through n (n is number of operators) are selected and available genes in the range of the selected points are shifted to the left rotary. Typical mutation has been presented in figure 5.

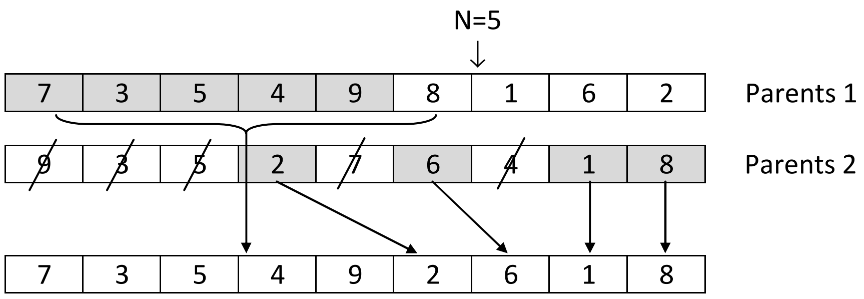

***Figure 4. Crossover operator for a typical system of Table 1***

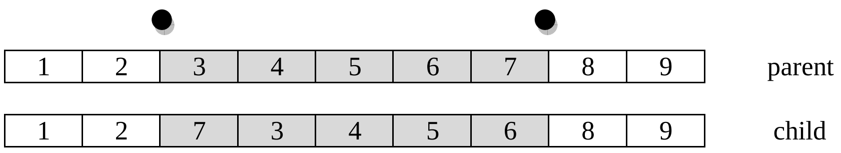

***Figure 5. Applying mutation operator for sample chromosome***

### *3.6 Tabu search*

In this stage, before selecting chromosomes for shifting to the next generation, the available population is optimized by tabu searching algorithm. Definition of proximity in the proposed tabu searching algorithm is defined based on shifting two genes, as total possible shifts of a chromosome genes are kept in a list called tabu list. Two genes are selected randomly from tabu list for shifting, and then one would be eliminated from tabu list to avoid repeated selection. Number of neighbor selection is equal to number of defined selections to optimize chromosome as much as possible. This trend would be applied to all existing chromosomes of population.

### *3.7 Chromosome selection*

Chromosome selection for next generation in our proposed algorithm is an integrated one. In selecting based on combinative method, first, chromosomes are ordered based on fitness and the repeated chromosomes are eliminated. Then 10% of those chromosomes with better fitness are selected and the remaining chromosomes are selected randomly for next generation of population.

### *3.8 Stop constraint*

In the proposed algorithm, constraint of stopping algorithm is bound to limiting the number of generation, meaning that if the number of generation reaches to regard number, algorithm stops.

## Implementation Result:

For implementation of the proposed algorithm, we used C#.Net 2008 programming language. Algorithm was applied to an Intel® Pentium ® processor 300 GHz and RAM 2.00 GB. Ten series of test data were defined to measure the effectiveness of our proposed algorithm. Designed test data set is called Test-j-o. In this naming policy, , j is the number of jobs and o is number of operation of each job in test data. Table 3 shows the designed test data.

***Table 3. Designed to test data***

| Test | Job count | Operation count | Initial population | Generation count |
|---|---|---|---|---|
| Test _4_4 | 4 | 4 | 70 | 200 |
| Test _5_5 | 5 | 5 | 70 | 200 |
| Test _6_6 | 6 | 6 | 70 | 200 |
| Test _7_7 | 7 | 7 | 70 | 200 |
| Test _8_8 | 8 | 8 | 70 | 200 |
| Test _10_10 | 10 | 10 | 70 | 300 |
| Test _15_15 | 15 | 15 | 70 | 400 |
| Test _20_10 | 20 | 10 | 100 | 400 |
| Test _20_20 | 20 | 20 | 100 | 400 |

Figure 1 illustrates algorithm of result of comparing proposed algorithm and DGA algorithm with data set of Test-7-7. In this data test there are 7 jobs and each job has 7 operations. Results show that the proposed algorithm has achieved better solution than the former DGA algorithm.

Figure 6 illustrates results diagram of comparing the proposed algorithm and DGA algorithm with data set of Test-15-5. In this data set there are 15 jobs wherein each job includes 15 operations. Results show that our proposed algorithm has found a better solution for big systems than the one proposed by DGA algorithm.

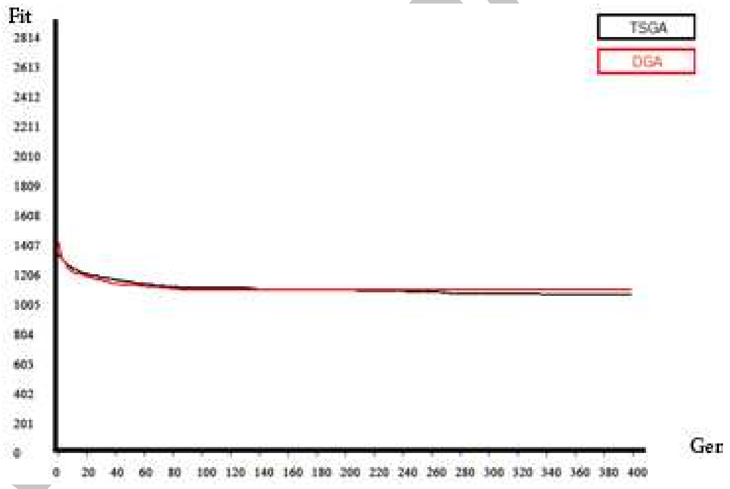

***Figure 6. The results compare the proposed algorithm and the algorithm for data Test_15_15 DGA***

Figure 7 and 8 show results of comparing proposed and DGA algorithms based on distribution diagram on data Test-7-7 and data Test-15-5. As can be seen in this figure, in our proposed algorithm population density has been kept stable, and using proper genetics operators leads to avoidance of forward divergence of solutions. This results in finding better solutions in proposed algorithm.

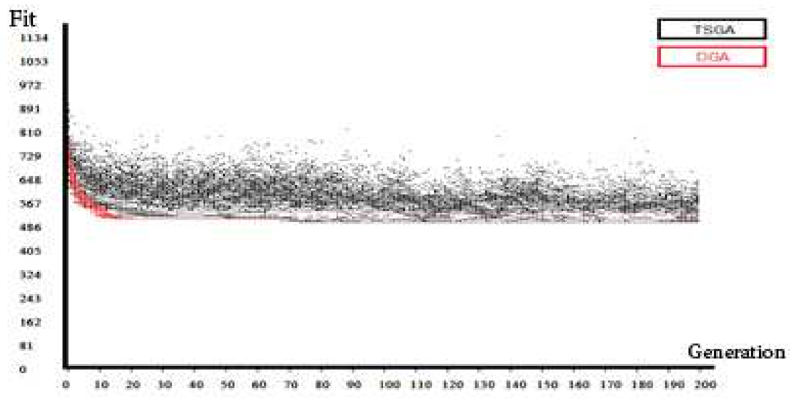

***Figure 7. Scatter diagram comparing the results of the proposed algorithm and the algorithm for data Test_7_7 DGA***

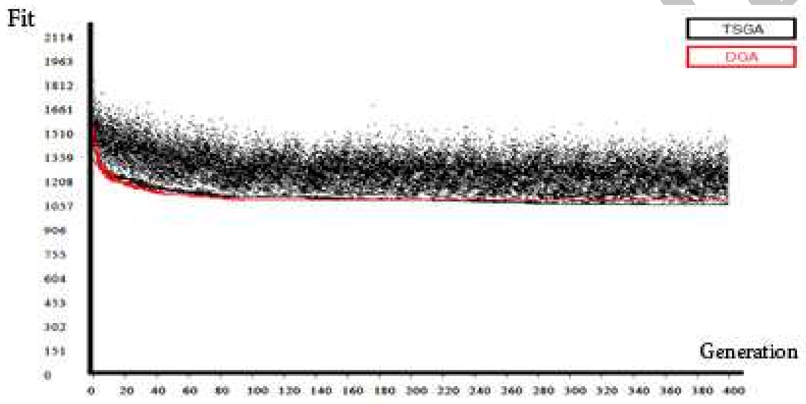

***Figure 8. Scatter diagram comparing the results of the proposed algorithm and the algorithm for data Test_15_15 DGA***

Table 4 shows the results from comparing the proposed algorithm with DGA algorithm. As it shows, our proposed algorithm needs more run time than DGA algorithm, but always attains more optimized solutions due to using tabu searching and the proper operators which increase chromosomes variety. Therefore, it can be concluded that our proposed algorithm has better efficiency than DGA algorithm.

*Table 4: The results compare the proposed algorithm and the algorithm DGA*

| Problem | TSGA | | | DGA | | |
|---|---|---|---|---|---|---|
| | The best fit | The best fit | The worst fit | Run Time | The best fit | The worst fit |
| Test_4 _4 | 7 | 287 | 696 | 5 | 287 | 710 |
| Test_5 _5 | 11 | 350، | 713 | 7 | 350 | 719 |
| Test_6 _6 | 19 | 436 | 882 | 11 | 436 | 837 |
| Test_7 _7 | 31 | 496 | 1128 | 18 | 506 | 1086 |
| Test_8 _8 | 48 | 578 | 1031 | 27 | 578 | 1048 |
| Test_10 _10 | 167 | 582 | 1276 | 99 | 585 | 1143 |
| Test_15 _15 | 850 | 1051 | 2104 | 848 | 1080 | 1906 |
| Test_20 _10 | 864 | 1281 | 2027 | 877 | 1281 | 1788 |
| Test_20 _20 | 4302 | 1314 | 2532 | 2554 | 1342 | 2338 |

## 4. Conclusion

In this paper, a new method is presented based on combination of genetics algorithm and tabu search for solving scheduling open shop problem.

This combination causes makes more efficient the searching performance. Also, using different genetics operators of proposed algorithm would keep up the diversity of chromosomes and also increased the algorithm efficiency in finding better solutions. Our empirical results show that the proposed algorithm has better efficiency compared with DGA algorithm and achieved better solutions, though has got a longer run time.

## 5. References

[1] Chinyao Lowa, Yuling Yeh, Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separate. Robotics and Computer Integrated Manufacturing 25 (2009) 314–322 .

[2] Gonzalez, S.; Sahni, T., Open Shop Scheduling to Minimize Finish Time, J. Assoc. of Comput. Mach., Vol. 23, 1976, 665 – 679.

[3] M. Pinedo, Scheduling, Theory, Algorithms, and Systems, prentice-Hall, Englewood Cliffs,NJ, 1995.

[4] P. Brucker, J. Hurink, B. Jurish, B. Wostmann, A branch and bound algorithm for the open-shop problem, Discrete Applied Mathematics 76 (1997) 43-59 .

[5] S.V. Sevast'janov, On some geometric methods in scheduling theory: A survey, Discrete Applied Mathematics 55 (1994) 59_82.

[6] Alcaide, D.; Sicilia, J.; Vigo, D., A Tabu Search Algorithm for the Open Shop problem, Top, Vol. 5, 1997, 283 - 286.

[7] Liaw, C.-F., A Hybrid Genetic Algorithm for the Open Shop Scheduling Problem, European J. Oper. Res., Vol. 124, 2000, 28 -42.

[8] Prins, C., Competitive Genetic Algorithms for the Open Shop Scheduling Problem, Math. Meth. Oper. Res. , Vol. 52, 2000, 389-411.