Solving the Multi-objective flexible job shop scheduling problem by the population-based algorithms

Zahra Rafie-Majd, Mohammad Mohammadi*, Bahman Naderi
*Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran
Corresponding author email: mohammadi@khu.ac.ir

Abstract

In this paper, the flexible job shop scheduling problem with the objective of optimization three objective functions simultaneously has studied. Due to the complexity of this problem exact methods cannot be used to find the optimum solution and therefore, heuristic and meta-heuristic approaches are suitable. In this research, Biogeography-based optimization (BBO) and Genetic Algorithms have been chosen among the population-based optimization algorithms. These algorithms were provided in the MATLAB programming language. Finally, the computational results obtained from the algorithms of this research were analyzed and several research fields have been suggested for the future studies.

Keywords: Flexible job shop scheduling problem, Simultaneous optimization, Biogeography-based optimization algorithm, Genetic Algorithm.

1. Introduction

Obviously, one of the main factors of competition is time and one of the most important success indicators of the projects is their scheduling. The classic job shop scheduling problem (JSSP), one of the most well known problems in the field of Scheduling, is a special case of the project scheduling problem with resource-constrained, where resources are just machines .In this problem there are a few machines and some jobs and the jobs must be processed on the machines .This problem is known as the most famous combinatorial optimization problems. JSSP consists of two sub-problem can be seen below :Routing and scheduling

Like many classic problems, job shop scheduling problem also has evolved over time and has come closer to real-world problems and some assumptions have been added to it. One of the most important of these assumptions is flexibility. Obviously, Flexibility in scheduling problems improve the system performance .In Flexible Job Shop Scheduling Problem (FJSSP) each operation is processed by a set of machines. In other words in a classic problem any action can be done by a machine and with a predetermined sequence ,while the first assumption is removed in a flexible mode and so for each operation a set of capable machines can be offered. Thus, in addition to sequencing jobs we face the additional problem that is allocating capable machines to each operation.it should be noted that in JSSP, different types of flexibility can be defined [1].

Another factor that is generally considered in optimization problems is about the number of objective functions .Most real-world optimization problems have more than one performance criteria (objective function) and the optimal solution achieve when all of these objective functions reach to their optimum value. These kinds of problems are called multi-objective optimization problems. Today, evolutionary algorithms are effective tools for solving multi-objective optimization problems .

Obviously, with regard to flexibility, the complexity of the problem will be greater .Since the job shop scheduling problem is known as a NP-hard problem So, FJSSP (as a generalization of the JSSP) belongs to the NP-hard problems class, too. Hence, heuristic and meta-heuristic methods such as biogeography-based optimization (BBO) algorithm have been widely used in many studies to solve these problems [1].

Dan Simon in 2008 [2], presented this algorithm Inspired by animal and bird migration between islands. In fact, bio-geography is the study of the geographical distribution of species [3].

The classic job shop scheduling problem and its derivatives have been the subject of research by many researchers in which each of them used heuristic or meta-heuristic methods according to the problem features. Among these methods we can mention: the use of particle swarm optimization (PSO) algorithm [4], genetic algorithm [5, 6], hybrid algorithms [7, 8, 9], simulation annealing [11 and 10], tabu search [12], bee colony
[13], etc. (To view a summary of researches done in recent years on the field of single-objective and multi-objective flexible job shop scheduling problem, see [12]).

Biogeography-based optimization algorithm has also been considered for solving the optimization problems, including JSSP, in recent years. Among the studies that have been conducted in this area [3, 14] can be noted.

In this research, solving the flexible job shop scheduling problem with the objective of optimization three objective functions by BBO and genetic algorithm is studied. The rest of this paper is organized as follows: In Section 2, we illustrate the problem of this study. Section 3 is devoted to the solution approach and to our developed meta-heuristic algorithms. Section 4 presents computational results and finally, the last section describes the conclusions and suggestions for future research.

2. Definition of problem of scheduling flexible flow shop

In FJSSP, It is assumed that n jobs are processed on m machines. For each job one or more operations are considered. This means that for every operation, related to a particular job, there is a set of capable machines that the operation can be performed on them with special processing time.

So, in this problem we are looking for: (1) assigning each operation to the most appropriate machine and (2) select the best sequence of operations on each machine, so that the certain objectives of the problem are provided. The most common objectives can be mentioned as the following:

- Minimize the completion time of the last operation (also called Cmax).
- Minimize the maximum work load on the machines.
- Minimize total work load on all machines.

In the mentioned problem, the following assumptions are satisfied [3]

1- Operations of each job have a fixed and predetermined order.
2- Jobs have the same priority
3- There is no priority restriction among operations of different jobs.
4- Jobs are released at time 0 and machines are available at time 0.
5- Move time between operations and setup time of machines are ignored.
6- At any specific time, only one job can be processed on each machine.
7- During the process, operations cannot be broken off. (In other words, operations are Non Preemptive[15])

The following figure is an example of the flexible job shop scheduling problem with 3 jobs and 4 machines. $o_{ij}$ represents the operation i from job j.

<table>
<thead>
<tr>
<th>FJSP</th>
<th>Processing times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
</tr>
<tr>
<td>J1</td>
<td>O1,1</td>
</tr>
<tr>
<td></td>
<td>O1,2</td>
</tr>
<tr>
<td></td>
<td>O1,3</td>
</tr>
<tr>
<td>J2</td>
<td>O2,1</td>
</tr>
<tr>
<td></td>
<td>O2,2</td>
</tr>
<tr>
<td>J3</td>
<td>O3,1</td>
</tr>
<tr>
<td></td>
<td>O3,2</td>
</tr>
<tr>
<td></td>
<td>O3,3</td>
</tr>
</tbody>
</table>

Fig. 1- an instance of FJSSP with 3 jobs and 4 machines [3]

3. The proposed algorithms for solving the problem

3.1 Genetic Algorithm
Genetic algorithm is an evolutionary approach which is performed on an initial population. Then, it chooses parents, applies operators on them to generate children (offspring), and evaluates the generated offspring. The goal of this algorithm is to successively produce better solutions by selecting the better ones of the existing solutions more frequently for recombination.

One of the most important applications of Genetic Algorithm is optimization of different problems, on various topics such as functions optimization, combinatorial optimization (such as the traveling salesman problem), machine learning, decision processes etc.

In the following, components of the proposed genetic algorithm are introduced.

### 3.1.1 Chromosome structure and initial population

Solution representation and decoding procedures play important roles in algorithm performance. In meta-heuristic methods, different representation can be used [16, 17]. A set of chromosomes is called a population. Our algorithm uses some points mentioned in the research done by Wang et al [18], which has been previously used by other researchers [3]. In this method, each chromosome consists of two strings of genes in which the number of genes is equal to the number of operations of all jobs. The upper string represents the operation sequence and the lower string show machines sequence that jobs will be performed on them respectively.

For example, in Figure 2 (plotted based on the data from Figure 1), the upper string represents that the sequence of operations is as follows (from left to right):

- The first operation of job 3, the first and second operations of job 1, the first operation of job 2 and so on. Lower string also shows allocation of jobs to machines. The first three genes (from the left) are related to job 1, the next two genes are related to job 2 and three remaining genes are related to job 3.

Now, in order to determine which operations will be performed on which machine select the operation from the upper string and then find peculiar machine to it from the lower string. For example in the following figure, The first operation of job 3 will be performed on machine 3 and similarly for the other operations, you can specify the machine specific to each operation.

<table>
<thead>
<tr>
<th>Operations sequence</th>
<th>Jobs allocation to machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 1 1 2 3 1 2 3</td>
<td>1 2 2 1 2 3 2 4</td>
</tr>
</tbody>
</table>

In order to generate the initial population, we produce chromosomes with the predetermined numbers (the number of chromosomes in the population, or population size, is denoted as pop-size).

### 3.1.2 Evaluation of each chromosome's fitness value

Before introducing the fitness function of each chromosome (In the case of FJSSP has several objective function), the fitness value of each chromosome is calculated for each of objectives.

For this purpose we act in this manner: the first operation (in this example the number 3) Processing starts at 0. Obviously, processing of second operation of job 3, will not be started until the first operation of this job is done completely. On the other hand, the third machine that is currently processing the first operation of job 3, cannot accept another operation until the end of processing this operation. So, each operation is processed at the earliest possible time of its capable machine. Following this decoding process, create an active schedule for each chromosome or solution of FJSSP [3] and then the Make-span, total workload, and maximum workload will be calculated based on the schedule. We use the method proposed by motaghedi-larijani et al in 2010 [19], so we minimize the following objective function:
\[
F = \frac{|f_1 - z_1^*|}{|z_1^*|} + \frac{|f_2 - z_2^*|}{|z_2^*|} + \frac{|f_3 - z_3^*|}{|z_3^*|}
\]  

(1)

Where \( z = (z_1^*, z_2^*, z_3^*) \) is the ideal vector. In order to have a good estimation of ideal vector, we have run a genetic algorithm for each individual objective function to obtain \( z^* \) vector. \( f_1, f_2, f_3 \) are also single objective functions that minimize make-span, maximum workload and total workload respectively.

After calculating the fitness value for all the chromosomes existing in the population, chromosomes are arranged from the best to the worst fitness. Then, certain percent of the best chromosomes (which have the best fitness) are transmitted to the next generation (It is called elitism).

### 3.1.3 The crossover operator

For selecting chromosomes for the crossover operator, tournament selection mechanism (on the objective function value) is used. Then only upper string is considered and one of the jobs select randomly. Offspring chromosome copies the locations of operations of the selected job from parent no.1 and receives the remaining operations from another parent. See following figure.

![Fig. 3- Crossover operator](image)

Then, put the lower string (Jobs allocation to machines) of one parent as lower string of offspring chromosome and thus a complete new chromosome is made from two parent chromosomes.

### 3.1.4 The mutation operator

In this operator, one gene is selected from lower string of one chromosome (which selected randomly from the population) and will be replaced with another authorized value. Consider following figure:

![Fig. 4- Mutation operator](image)
Fourth gene (operation 1 from job 2) is selected randomly. Hence this operation can only be performed on machine 1 and machine 4, so the fourth gene, number 1, can only be converted to number 4.

3.2 Biogeography-based optimization algorithm

As stated previously, bio-geography is the study of the geographical distribution of species. Islands (or Habitat), the equivalent of chromosome in Genetic algorithm, which are appropriate places for settlement of geographical species, have high Habitat suitability Index (HSI), the equivalent of chromosome's fitness value in Genetic algorithm. In other words, High-HSI habitat represents a good solution and low-HSI habitat represents a poor solution [3]. As well chromosomes are composed of genes each HSI is composed of several SIV.

Since there are many similarities between this algorithm and genetic algorithm (see [3]), and BBO just like GA is a population-based algorithm, then Solution representation and decoding procedures and elitism of BBO algorithm is just like genetic algorithm, presented in Section 3.

In BBO, solution features emigrate from high-HSI habitats (emigrating habitat) to low-HSI habitats (immigrating habitat). Therefore, the migration operators, which are emigration and immigration, are used to improve and evolve a solution to the optimization problem. The rates of these two types of migration are also \( \lambda_i \) and \( \mu_i \) [2, 3].

\[
\lambda_i = I (1 - \frac{k_i}{n}) \\
\mu_i = E \left( \frac{k_i}{n} \right)
\]

In Eq. 2 and 3, \( k_i \) represents the rank of the ith habitat after sorting all habitats according to \( s \) and \( n \) represents the pop-size. Obviously, each habitat has its own emigration and immigration rates. \( E \) and \( I \) are also Input parameters of the algorithm and are mostly set to 1 [for more information, see 2, 3].

In each iteration of the algorithm, we should check that whether the specific habitat needs to be changed or not? (poor solution/low-HIS habitat). For this purpose, a random number between 0 and 1 is produced, and then is compared with \( \lambda_i \) of mentioned habitat, if the random number was smaller than \( \lambda_i \), the habitat is selected to improve.

The second step is to determine the high-HIS solution by implementation Tournament selection mechanism on \( \mu_i \).

After choosing these two habitats, migration operator applies on them just like crossover operator of genetic algorithm and therefore new habitat is created.

In this algorithm the mutation operator is used for increasing diversity among the population. In this case, a random number between 0 and 1 is selected and compared with the mutation probability, \( m_i \), (Equation 4).

\[
m_i = m_{max} \left( 1 - \frac{p_i}{p_{max}} \right)
\]

In this Eq., \( m_{max} \) denotes the largest amount of mutation rate and is determined by the user. \( p_i \) (in some references \( p_s \)) is the probability of existence of \( S \) species in the habitat [for more details please see 2 and 3].

4. Computational results

After developing and performing an algorithm, it is time to examine its efficiency in the intended area. For this purpose we use “job shop scheduling problem Library”

(see: [http://www.idsia.ch/~monaldo/fjsp.html#ProblemInstances](http://www.idsia.ch/~monaldo/fjsp.html#ProblemInstances)).

Figures 5 and 6 respectively, show GA and BBO performance in single-objective mode and about problem mk01. BBO algorithm parameters considered are as follows:

- \( m_{max} = 0.001 \), Pop-size=100, \( E = 1, I = 1 \)

- Genetic algorithm parameters are:
  - Pop-size=100, mutation rate=0.3
The stop criteria for both algorithms are 10 seconds.

Fig. 5- GA performance: (a) minimize make-span, (b) minimize maximum work load, (c) minimize total work load
Fig. 6- BBO performance: (a) minimize make-span, (b) minimize maximum work load, (c) minimize total work load

In order to compare the efficiency of the proposed algorithms of this research and future studies, computational results obtained from implementing the proposed algorithms on single-objective (table 1) and three-objective (table 2) modes is shown for 5 instances problem.

In three-objective case, pop-size is considered 10 and stop criteria is considered n (number of operations of all jobs) seconds. As mentioned before $z_1^*, z_2^*$ and $z_3^*$ is calculated from running algorithms for each individual objective function.

Table 1. The numerical results of problems (single-objective mode)

<table>
<thead>
<tr>
<th>Example no.</th>
<th>Problem size - number of operations</th>
<th>The results related to the BBO (single objective)</th>
<th>The results related to the GA (single objective)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Minimizing make span</td>
<td>Minimizing the total workload</td>
</tr>
<tr>
<td>MK01</td>
<td>$10^5 - 25$</td>
<td>53</td>
<td>10.0621</td>
</tr>
<tr>
<td>MK02</td>
<td>$10^5 - 38$</td>
<td>59</td>
<td>10.0933</td>
</tr>
<tr>
<td>MK03</td>
<td>$15^4 - 150$</td>
<td>271</td>
<td>10.4831</td>
</tr>
</tbody>
</table>

Table 2. The numerical results of problems (multi-objective mode)

<table>
<thead>
<tr>
<th>Example no</th>
<th>The results related to the BBO (multi-objective)</th>
<th>The results related to the GA (multi-objective)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The value of objective function</td>
<td>Calculation time (s)</td>
</tr>
<tr>
<td>MK01</td>
<td>0.1067</td>
<td>180.3840</td>
</tr>
<tr>
<td>MK02</td>
<td>0.1423</td>
<td>180.5712</td>
</tr>
<tr>
<td>MK03</td>
<td>0.0922</td>
<td>241.0371</td>
</tr>
<tr>
<td>MK04</td>
<td>0.1750</td>
<td>240.5067</td>
</tr>
<tr>
<td>MK05</td>
<td>0.1533</td>
<td>120.8072</td>
</tr>
</tbody>
</table>

As can be seen, in the single-objective mode there is no noticeable difference between these two algorithms, but in multi-objective modes, BBO in most cases has better performance.
5 Conclusion and suggestions for further studies

In this paper, we study the flexible job shop scheduling problem with three objectives: minimize the make-span, total workload, and maximum workload. Since this problem cannot be solved by exact methods therefore for solving this problem, we applied meta-heuristic algorithms (BBO and GA). Structures of the algorithms and used operators have been described and finally some instance problems from job shop scheduling problem library were selected and have been solved with the help of algorithms. The findings show the superiority of BBO on the genetic algorithm in multi-objective problems.

In future researches, the performance of these algorithms can be improved by combination these algorithms and making a hybrid algorithm. Due to the inherent uncertainty of the scheduling problems, fuzzy logic can be an interesting field for future studies.

References