Multi objective Flexible Job Shop Scheduling Optimization Using BFOA

Nagendra Deshmukh¹, Anoop Kumar Sood²
¹,²Department of Manufacturing Engineering, NIFFT, Hatia-834003, Ranchi, Jharkhand, India

Abstract—This study proposes an adaptive bacterial foraging optimization algorithm (ABFOA) for solving multiobjective flexible job-shop scheduling problem (FJSP). Bacterial foraging optimization algorithm (BFOA) is a new bio-inspired evolutionary computational technique, which is inspired by foraging behavior of E.coli bacterium present in the human intestine. To make it suitable in accordance to combinatorial nature of FJSP some modifications are proposed, in terms of bacteria representation and their search strategy, for faster convergence of bacteria towards Pareto optimal solution. The computational results prove the effectiveness of the proposed ABFOA for solving flexible job-shop scheduling problem.

Keywords— Adaptive Bacteria Foraging Algorithm (ABFOA), Bacteria Foraging Optimization Algorithm (BFOA), Flexible job shop scheduling (FJSP), Multiobjective optimization, Pareto optimal solution.

I. INTRODUCTION

The flexible job-shop scheduling (FJS) is a branch of production scheduling, which is NP-hard combinatorial optimization problem [1]. The FJS problem (FJSP) is composed of two set of problems namely (1) to determine assignment of operations to a machine out of capable machines and (2) sequencing of operations in assigned machines; so as to satisfy some pre-defined objectives. Most often in production environment there are number of conflicting objectives and to find the optimum schedule satisfying all these objectives is a challenging task [2]. In this direction, applications of evolutionary algorithms like genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO) to solve the multi objective FJSP provided competitive results and are found more appropriate and efficient as compared to other optimization techniques [3, 4, 5].

Bacterial Foraging Optimization Algorithm (BFOA) is a new bio-inspired, self-organizing type of evolutionary algorithm which mimic the foraging behavior of E.coli bacterium present in the human intestine. In the past few years, BFOA has been successfully used to solve continuous search space problems [6].

However, studies and research focused on using BFOA for solving combinatorial and NP-hard discrete problems is scarce and not been tried to solve FJSP. The way the bacteria explore and exploit the nutrient rich region makes this algorithm an excellent candidate in solving multi objective flexible job shop scheduling problem (MOFJSP). To make this algorithm suitable to combinatorial nature of FJSP and eliminate the complexities of original BFOA, some modifications are proposed, which enhance the convergence characteristics of the algorithm.

II. PROBLEM DESCRIPTIONS AND OBJECTIVES

Multi objective FJSP addressed in this study is defined as follows:

- There are set of $n$ jobs $J=\{J_1, J_2, \ldots, J_n\}$, indexed by $i$.
- There are set of $m$ machines $M=\{M_1, M_2, \ldots, M_m\}$, indexed by $k$.
- Each job $i$ has total $Q_i$ operation and the operation sequence is given by $O_{ij}$ for $j=1, 2, \ldots, Q_i$.
- For each $O_{ij}$ there is a set of machines capable of performing it, represented by $M_{ijk}$.
- If $M_{kij} \subseteq M$ it is partial flexible job shop scheduling problem (p-FJSP). Each operation could be processed on one machine of subset of $M$. If $M_{kij} = M$ then it has total flexibility, it is total FJSP (t-FJSP). Each operation could be processed on any machine of $M$.
- Processing time of an operation $O_{ij}$ on machine $k$ is predefined and given by $P_{ijk}$.

The objective is to determine an assignment and a sequence of the operations on the machines so that some objectives are satisfied. In this study following three objectives are minimized:

1. Time required for all jobs to be processed according to a given schedule that is makespan $F_m$.

$$F_m = \max_{i=1}^{Q} \{ C_{i,Q} \}$$

458
Where, $C_{i,Q_i}$ is completion time of all operation of $i^{th}$ job.

2. Total workload of machines ($F_i$), which is of interest assigning the machine with relatively small processing time for each operation to improve economic efficiency.

$$F_i = \sum_{k=1}^{m} \sum_{j=1}^{n} P_{ij}x_{ijk}$$  \hspace{1cm} (2)

3. The maximal workload ($F_m$), which considers the workload balance among all machines to prevent too much work being assigned to a single machine.

$$F_m = \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{n} P_{ij}x_{ijk} \right\}$$ \hspace{1cm} (3)

Subject to:

$$C_{ij} - C_{i(i-1)} \geq P_{ij}x_{ijk}$$  \hspace{1cm} (4)

$$j = 2,3,...,Q_i; \forall i,k$$

$$\sum_{k \in M_{ij}} x_{ijk} = 1, \forall i, j$$  \hspace{1cm} (5)

Hypotheses considered are summarized as follows:

- All machines are available at time zero.
- Machines never breakdown and always available.
- All jobs are available and ready to start at time zero.
- All jobs are independent to each other
- Each machine can process only one operation at a time (resource constraint).
- Each operation can be processed without interruption (non-preemption condition) on available machines.
- Recirculation occurs when a job could visit a machine more than once.
- The order of operations for each job is predefined and cannot be modified (precedence constraint).
- The setup time of any operation is independent of the schedule, fixed and included in the corresponding processing time.

The study adopted Pareto-optimal concept to solve MOFJSP. The optimality notion in the Pareto approach can be formulated as follows:

- The Pareto-optimal set is constituted of non-dominated solutions.

A solution is non-dominated, if it is not dominated by any other one. In our problem, we define that a schedule $x$ dominates ($\prec$) a schedule $y$ by

$$x \prec y \iff \forall i \in 1,2,3 \ f_i(x) \leq f_i(y)$$
$$\exists i \in 1,2,3 \ f_i(x) < f_i(y)$$

III. BACTERIAL FORAGING OPTIMIZATION ALGORITHM (BFOA)

BFOA is based on foraging strategy of E.coli bacteria present in human intestine [7]. These bacterium, for their existence, try to find a nutrient rich region in there multi dimension search space. Initially, bacterium measures the nutrient concentration at present location, swim to same direction or tumble to random direction and measure the concentration there. This is known as Chemotaxis behaviour of E.coli bacteria and is limited by the lifespan of the bacterium. Bacteria while being at richest food location (optimum bacterium) also try to attract other bacteria so that they converge to the optimum region more rapidly [8]. Bacteria which have better foraging strategy survives through evolution based reproduction in which the best set of bacteria gets divided into two cell. The worst half of bacteria, which are having poor foraging behaviour, gets eliminated and are replaced with healthier half. Sometimes, there can be such uncertain events occur which disturb the evolution process and cause the elimination of the set of bacteria or a group is dispersed into a new environment. This may destroy the smooth operation or may place the bacteria near good nutrient zone. Hence, the elimination and dispersal is also consider as a part of the motile behaviour of E.coli bacteria. Finally, BFOA is categorized into four stages: Chemotaxis, Swarming, Reproduction and Elimination-Dispersal.

A. Chemotaxis

The motile behavior of E.coli bacteria completes in two ways i.e., it can run (swim) to a fixed length or it can tumble in different direction. With Swimming and Tumbling bacteria complete the chemotactic step where they try to move towards nutrient gradient and avoid noxious region. Suppose the position of $i^{th}$ bacteria at the $j^{th}$ chemotactic step, $k^{th}$ reproduction step, and $l^{th}$ elimination-dispersal event be $\Theta^i(j,k,l)$. Then the direction of movement of bacteria after tumble can be computationally represented by:

$$\Theta^i(j+1,k,l) = \Theta^i(j,k,l) + C(i)\phi(j)$$  \hspace{1cm} (6)
Where, $C(i) > 0, i = 1,2,\ldots,S$ denote a basic chemotactic step size and $\phi(j)$ represents a tumble (a unit random direction). $S$ denotes the total number of bacteria i.e. total population size.

**B. Swarming**

Bacteria while being at nutrient rich region, also have cell-to-cell attraction to converge altogether in a group at the richest food location. It also repels the nearby cells such that it does not consume the nearby nutrients and it is not physically possible to have two cells at the same position. To consider this swarm characteristics of the bacteria, a penalty function is added to the original function which is time varying and based on the relative distance of the individual bacterium with the global optimum bacterium. The cell-to-cell signaling can be represented by:

$$
F_{cc}(\theta_g(j,k,l),\theta(j,k,l)) = \sum_{i=1}^{m} F_{cc}^i(\theta_{gm}(j,k,l),\theta(j,k,l))
$$

$$
= \sum_{i=1}^{m} \left[ -d_{\text{attract}} \exp \left( -w_{\text{attract}} \sum_{a=1}^{n} (\theta_{gm} - \theta_{attract})^2 \right) \right]
+ \sum_{i=1}^{m} \left[ h_{\text{repellent}} \exp \left( -w_{\text{repellent}} \sum_{a=1}^{n} (\theta_{gm} - \theta_{repellent})^2 \right) \right]
$$

(7)

Where $F_{cc}$ is the penalty function to be added to the actual function, termed as swarm attractant cost, $\theta_{gm}$ is the position of global optimum bacterium, $S$ is the total population, $m$ represents the $m^{th}$ parameter of bacterium location, $d_{\text{attract}}$ and $w_{\text{attract}}$ represent the depth and width of the attractant signal respectively, $h_{\text{repellent}}$ and $w_{\text{repellent}}$ are the depth and width of the repellent signal respectively. Since, it is physically not possible to have two cell at same location so it is assumed that $h_{\text{repellent}}=d_{\text{attract}}$.

**C. Reproduction**

After the chemotactic step, bacteria who have had sufficient nutrients will reproduce i.e., split into two cells. In our problem, population is sorted depending upon the fitness value of each bacteria. The healthier half which have minimum cost value survives and reproduce whereas the bacteria which do not have much nutrient i.e. not as healthy gets eliminated. Thus this stage ensures the fittest bacterium to explore more and keep a constant population size. Arrange the population in ascending order of accumulated cost.

The accumulated cost ($F_{sw}$) is calculated as the sum of cost function value ($F$) and swarm attractant cost ($F_{cc}$) which can be computationally given by:

$$
F_{sw}(i,j,k,l) = F(i,j,k,l) + F_{cc}(\theta_g(j,k,l),\theta(j,k,l))
$$

(8)

The healthier half population (the bacteria which have minimum accumulated cost value) will survive and reproduce at the same location and the other worst half are eliminated and are replaced by randomly generated new population.

**D. Elimination-dispersal**

Sometimes during evolution process, there can be such event occur where the group of bacteria are killed or disperse into entirely a new environment due to uncertain change in the local environment where the population of bacteria lives. This elimination-dispersal event may destroy the continuous chemotactic step in progress, but it may also disperse the bacteria in the new environment of rich nutrients. So this stage assure the algorithm, not to trap into local optima but to explore it towards the global optima over the entire multidimensional search space and is consider as a part of population level mobile behaviour of bacteria.

**IV. ADAPTIVE BACTERIA FORAGING OPTIMIZATION ALGORITHM (ABFOA)**

In the present study objective is to schedule a set of jobs on a set of machines for FJSP. Here, the solution space is discrete. The original BFOA is developed to solve various continuous search space problem but the study is scarce in the field of using BFOA in scheduling problems. BFOA has given good results on solving the continuous search space problem. Hence, it may possible to find a set of Pareto optimal solution of FJSP by using this evolutionary algorithm in acceptable time. With this prospect, present section will discuss the modification made in BFOA to achieve the predefined objectives.

To understand the algorithm more easily, we designed an example of flexible job shop scheduling problem shown in Table I, which will be used throughout the paper. In Table I, there are $3$ jobs and $4$ machines, where rows and column represents operations and machines respectively and the processing time of each operation on the corresponding machine is shown. The machines which does not belong to alternative machine set of the operation are denoted by "-" on the cell. So this is a case of partial FJSP.
Bacteria representation (Coding)

The first important task for successful application of BFOA to solve multiobjective FJSP is bacteria coding. In this paper we divided the bacteria into two parts, the length of each part is same and it is equal to the sum of all operations of all jobs. The first part defines the sequence of operation on all machines and the second part defines the routing problem that assigns each operation to a machine selected out of a set of capable machines. Both sequences are read from left to right. The bacteria representation for one possible encoding of operation selection and machine selection is shown in Fig 1.

A. Bacteria representation (Coding)

<table>
<thead>
<tr>
<th>Job</th>
<th>operation</th>
<th>M₁</th>
<th>M₂</th>
<th>M₃</th>
<th>M₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>J₁</td>
<td>O₁₁</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>O₁₂</td>
<td>-</td>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>J₂</td>
<td>O₂₁</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>O₂₂</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>O₂₃</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>J₃</td>
<td>O₃₁</td>
<td>8</td>
<td>-</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>O₃₂</td>
<td>-</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

TABLE I
PROCESSING TIME TABLE OF A P-FJSP INSTANCE.

A good initial bacteria, operation sequence and machine selection is shown in Fig 2.

B. Bacteria decoding

Each encoded bacteria needs to be decoded into the active schedule [9]. Bacteria is scan from left to right. The first part is the operation sequence part and second is the machine selection part.

1. Operation sequence: In operation sequence part, each job number occurs n times (n total number of operation associated with this job). The mth occurrence of each job represents the mth operation of this job and the repetition of number of each job represents the order in which the operations of jobs are scheduled and is shown in Fig 2.

2. Machine Selection: In the machine selection part, each integer value shows the index of the array of alternative machines set of each operation of all the jobs. As per the structure in Fig.1. M₁ is selected to process operation O₁₁. The value can be 2 or 3 since operation O₂₁ can also be processed on M₂ or M₃. This provides enough flexibility to encode the FJSP and hence improves the search process in such discrete solution space problem.

To generate a schedule, assign each operation to a predetermined machine as per the operation sequence array. For jth job, jth operation, determine its processing time in all machines. Further, determine minimum time that the machine can process the operation. Check for the unallocated positions between the already scheduled operations for that machine from left to right. If unallocated space is equal to the processing time of the operation Oᵢₗ, then schedule Oᵢₗ at that space, otherwise it would be scheduled at the end.

C. Initial population

For the success of any evolutionary approach, population initialization plays a key role. A good initial bacteria provides a diverse search space to avoid convergence of solution towards local optima. In this study, to generate a good initial bacteria, operation sequence are assigned randomly. The machines are assigned by considering maximum workload and operation time of machine on which operation have already been performed at once.

D. Global best bacterium

For selecting global best bacterium, Bacteria are arranged in ascending order of each to objective by using tournament selection approach. Tournament selection is based on non-dominated sorting of solution based on objective values. The crowding distance, distance between the non-dominated solution in Pareto optimal front, is calculated computationally as:

\[
d_b = \sum_{b=1}^{s_f} \left[ \sum_{j=1}^{3} \frac{F_{j(b_{k+1})} - F_{j(b_k)}}{F_{j}^{max} - F_{j}^{min}} \right] \quad \text{if } b = 1 \text{ or } b = \text{last}
\]

\[
d_b = \sum_{b=1}^{s_f} \left[ \sum_{j=1}^{3} \frac{F_{j(b_{k+1})} - F_{j(b_k)}}{F_{j}^{max} - F_{j}^{min}} \right] \quad \text{otherwise}
\]
Where, $S_i$ is total number of bacteria in a front, $F_{j_g}$ is the value of the $j^{th}$ objective of $S_b$ bacterium, $F_{j}^{\text{max}}$ and $F_{j}^{\text{min}}$ are the maximal and minimal values of the $j^{th}$ objective. The global best bacterium is selected based on the solution which have minimum rank and maximum crowding distance.

The basic idea behind the crowing distance is finding the Euclidian distance between each individual in a front based on their objective values. Once the rank and crowding distance is assigned to each individual, global best bacterium is selected based on minimum rank and maximum crowding distance. If more than one bacterium satisfies this condition then all these bacteria can be considered as best but in this study out of these bacteria one bacterium is selected randomly as global best bacterium.

### E. Swarm attractant cost

Swarm attractant cost is the penalty cost added to the objection function and is proportional to the relative distance of individual bacteria from the global optimum bacteria. We are considering that, the global optimum bacterium gives the best operation sequence as per the desired objective. So the relative distance is measured by calculating difference between the sequences of each bacteria with the sequence of global optimum bacterium. The swarm attractant cost function is calculated computationally as:

$$
\sum_{m=1}^{l} (\theta_m^g - \theta_m^b)^2 = \frac{\sum_{m=1}^{l} \sum_{i=1}^{k} (p_m^g - p_m^b)^2}{\sum_{i=1}^{k} \sum_{m=1}^{l} (p_m^g - p_m^b)^2} \quad (10)
$$

Where, $\theta_m^g$ is the global best bacterium in $m^{th}$ parameter, $l$ is the length of the operation sequence part. $P_m^g$ and $P_m^b$ are the position of $O_j$ operation of global best bacterium and $\theta_m^b$ bacterium respectively.

### F. Swimming

Bacteria swim or tumble depends upon the fitness function value. It swims (run) to a fixed length if it is in desired direction. Hence, there is a small change in the local environment of the bacteria. To adapt this change, mutation is applied to the bacteria. The mutation can be seen as a perturbation to the bacterium during the evolutionary process.

During last decade, several mutation operators have been introduced for permutation representation including inversion, insertion, displacement, reciprocal exchange mutation and shift mutation [9]. Out of several mutation operators, inversion mutation operator is adopted. In inversion mutation, two positions within an operation sequence substring are selected at random and substring is inverted between these two positions. Inversion mutation of operation sequence part may make the machine sequence substring infeasible especially in the case of partial flexibility. To rectify this, machines sequence substring is checked for feasibility. If at a particular location inappropriate machine is assigned then it is replaced by machine chosen randomly from the set of available machines.

### G. Tumble

Tumble of bacteria means the movement of bacteria in altogether random direction. This motile behaviour of bacteria can place the bacteria to an entirely new region where the fitness is unknown. This is similar to the crossover operator used in GA which results in generation of new chromosome (child) from the two parent chromosomes, whose fitness is unknown at the time of birth. Based on same analogy tumble is substitute by the crossover operation of bacteria.

In this study, for the tumble of bacteria, we apply precedence preserving order-based crossover (POX). Like the mutation operator, there exist a number of crossover operators in the GA literature [10]. Operation sequence may be either infeasible in the sense that some precedence constraint are violated or illegal in the sense that the repetitions of some symbols are not equal to prescribed numbers. To prevent this precedence preserving position based crossover (POX) has been adopted for operation sequence sub string in present study. In the POX crossover, the two jobs are randomly selected, then produce the child by copying these jobs at the position at which they occurs in the first parent. Now, delete these jobs from all the positions at which they occur in parent two. Orderly fill the empty position in child with the alleles of parent that belong to in their previous sequence. Based on the operation sequence of child, new machine sequence is generated by selecting machine from the set of feasible machines for that particular operation randomly.

The other part of the adaptive bacterial foraging algorithm (ABFOA) follows the original BFOA structure. In first step bacteria swim in preferred direction and after that decision is made to swim or tumble based on swarm attachment cost.
For simplicity of the computation, the Reproduction and elimination-dispersal steps are combined and consider as a single step where the set of Pareto optimal solution are retain and the worst half is eliminated. The population of bacteria is maintained by generating new population randomly.

The detail steps in ABFOA are given as follows

Step-1: Initialize algorithm parameters

Step-2: Generate initial solutions

Step-3: Perform non dominant sorting and calculate crowding distance.

Step-4: Determine global best bacteria

Step-5: Calculate swarm attachment cost

Step-6: Perform step3 with swarm attachment cost as objective function

Step-7: Allow bacteria to swim in preferred direction.

Step-8: Repeat step 4 to step 6 for new bacteria positions.

Step-9: Compare swarm attachment cost calculated in step 5 and Step8.

Step-10: Repeat step 3

Step-11: Retain the bacteria whose rank is one and generate remaining bacteria randomly.

Step-12: Stop if stopping criteria is met else go to step3.

V. COMPUTATIONAL RESULTS OF ABFOA

Computational experiment are carried out to illustrate the effectiveness of proposed adaptive bacteria foraging algorithm. The two problems have been taken from Kacem et al. [1], whose scale is $n \times m$ ($n$ = number of jobs, $m$ = number of machines). The first is 4-job and 5-machine problem and second is 10-job and 10-machine problem. For both the problems population is 200 and maximum iterations are 100. Parameters of swarming such as $d_{attract}$, $w_{attract}$, $h_{repel}$, and $w_{repel}$ are set at 1.9, 0.2, 1.9, and 10, respectively. These problem instances are solved by ABFOA and the results are compared with results obtained by other researchers.

The best solutions obtained by proposed approach ABFOA for kacem et al. $4 \times 5$ is shown in TABLE II and it is compared it with all others algorithm considered in the literature. From TABLE II, computational results shows that the best solution obtained by ABFOA dominate the solution obtained by Kacem et al. (ALC-GA) and also reached the best solution obtained by all other algorithms considered in the literature.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_M$</td>
<td>16</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>$W_M$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$W_T$</td>
<td>34</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>
In this study, we proposed an adaptive bacteria foraging algorithm (ABFOA) for solving multi objective FJSP i.e. minimization of makespan, maximum workload and total workload. It is clear from results that the solution obtained by proposed algorithm reached the optimal solutions as obtained by other algorithms of the literature. The proposed approach is flexible enough to be used for both total-FJSP and Partial-FJSP. In the future, firstly we will work on reducing computation time. Further, we will work on using adaptive BFOA for solving more realistic flexible job shop scheduling problems.

VI. CONCLUSION

In TABLE III, we compare the minimum makespan obtained by all algorithms with ABFOA for 4 x 5 and 10 x 10 Kacem et al.’ instances. Form the results, we find that proposed ABFOA reached the minimum makespan as obtained by all other algorithms. The set of Pareto optimal solution (S-1, S-2, S-3, and S-4) obtained by ABFOA for both problems are shown in TABLE IV.

REFERENCES


