An Enhanced Bio-Stimulated Methodology to Resolve Shop Scheduling Problems

Divya. P, Narendhar. S and Ravibabu. V

Department of Computer Applications, School of Computer Science and Engineering, Bharathiar Universiy, Coimbatore, Tamil Nadu, INDIA

Abstract

This paper symbolizes the efficiency of Customized Bacterial Foraging Optimization algorithm. In this research work, Bacterial Foraging Optimization was combined with Ant Colony Optimization and a new technique Customized Bacterial Foraging Optimization for solving Job Shop Scheduling, Flow Shop Scheduling and Open Shop Scheduling problems were suggested. The Customized Bacterial Foraging Optimization was tested on the Benchmark instances and randomly created instances. From the implementation of this research work, it could be observed that the proposed Customized Bacterial Foraging Optimization was effective than Bacterial Foraging Optimization algorithm in solving Shop Scheduling Problems. Customized Bacterial Foraging Optimization can also be used to resolve real world Shop Scheduling Problems.

Keywords

Ant Colony Optimization (ACO), Bacterial Foraging Optimization (BFO), Job Shop Scheduling Problem (JSSP), Flow Shop Scheduling Problem (FSSP), Open Shop Scheduling Problem (OSSP), Customized Bacterial Foraging Optimization (CBFO)

1. Introduction

1.1 Ant Colony Optimization

ACO algorithm first proposed by M. Dorigo, in 1992 [40]. It is a metaheuristic in which a colony of ants capable of finding shortest trail from their nest to food sources using pheromone examinations. Real ants are not only capable of finding the shortest path from a food source to the nest as shown in the Figure 1 (Colorni et al., 1993; Dorigo and Gambardella, 1997; Holldobler and Wilson, 1990) without using visual prompts, but also they are competent of adapting to changes in the environment The probability that the ants coming delayed choose the path is proportional to the amount of pheromone on the path, earlier dropped by other ants. For example, they will get a new shortest path once the previous one is no longer possible.

1.2 Bacterial Foraging Optimization

BFO was introduced by Kevin M. Passino in 2000 for distributed optimization problems [9]. Bacterial Foraging Optimization (BFO) algorithm is a novel evolutionary calculation algorithm suggested based on the foraging activities of Escherichia coli (E. coli) bacteria living in human intestine [19]. The BFO algorithm is a biologically enthused computing method which is supported on mimicking the foraging activities of E. coli bacteria.

DOI : 10.5121/ijasa.2013.1203

The BFO Algorithm associates to the field of Bacteria Optimization Algorithms and Swarm Optimization. BFO algorithm is successfully applied in several real world problems and improved BFO metaheuristics were applied to optimization problems.

Framework for BFO algorithm

- Input the bacterial foraging parameters and independent variable, then specify inferior and superior limits of the variables and begin the elimination-dispersal steps, reproduction and chemotactic.
- Generate the locations of the independent variable arbitrarily for a population of bacteria. Estimate the intention value of each bacterium.
- Change the position of the variables for all the bacteria with the tumbling or swimming procedure .Perform reproduction and elimination procedure.
- If the maximum number of chemotactic, reproduction and elimination-dispersal steps is achieved, then output the variable corresponding to the overall best bacterium; Otherwise, do again the procedure by changing the position of the variables for all the bacteria with the tumbling /swimming procedure.

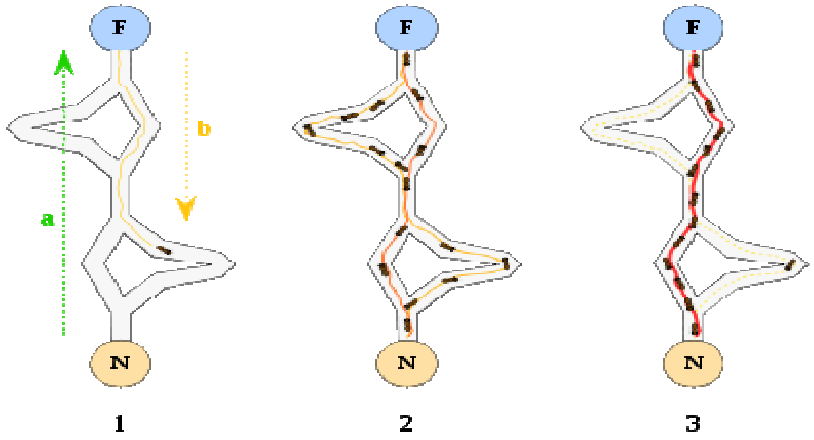

Figure 1: Double Bridge Experiment

1.3 Job Shop Scheduling Problem (JSSP)

Job	:	A piece of work that goes through series of operations.
Shop	:	A place for manufacturing or repairing of goods or machinery.
Scheduling	:	Decision process aiming to deduce the order of processing.

The JSSP is an operation sequencing problem on multiple machine subject to some precedence constraints among the operations. The JSSP can be explained as a set of n jobs represented by J_j where j =1,2…n which have to be processed on a set of m machines represented by M_k where k =1,2….m. Operation of j^{th} job on the k^{th} machine will be represented by O_{jk} with the processing time p_{jk} [25] .Each job should be processed through the machines in a exacting order or also known as technological constraint. Once a machine begins to process a job, no disruption

is allowed. The time required for all operations to complete their processes is called makespan. JSSP are widely known as NP-Hard problem. Figure 2 represents flow of JSSP.

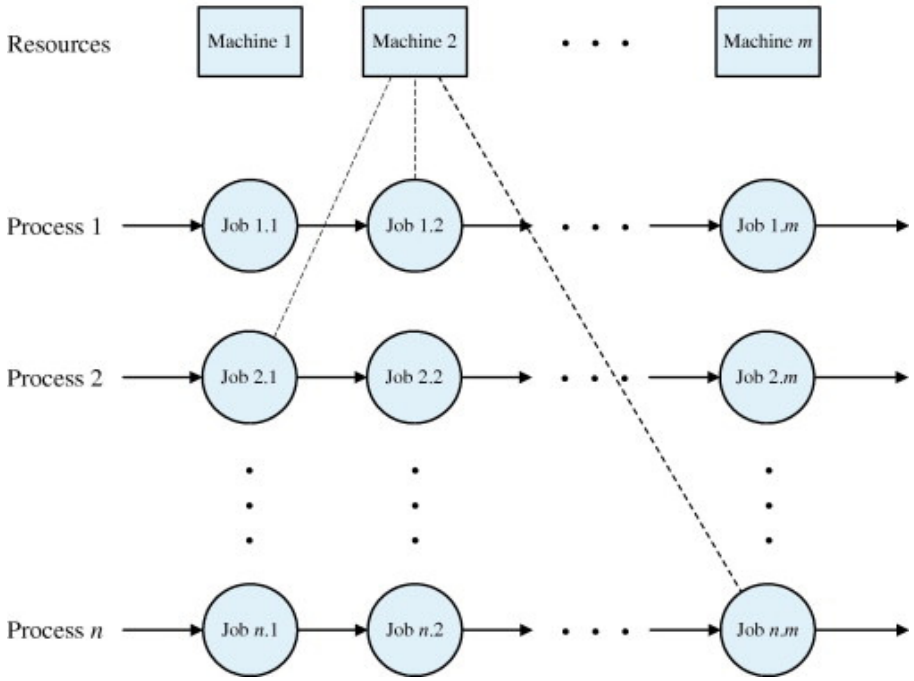

Figure 2: Flow of JSSP

Constraints

The JSSP focuses to two constraints, they are

- **Operation precedence constraint**

  The operation precedence constraint on the job is that the arrangement of operations of job is fixed and the processing of an operation cannot be interrupted and concurrent.

- **Machine processing constraint**

  The machine processing constraint is that only a solitary job can be processed at the identical time on the identical machine.

The main factor affecting to the JSSP is the nature of job shop. In static and deterministic job shop, all jobs are obtainable for processing without delay, and no fresh jobs appear over time. In the dynamic probabilistic job shop, jobs arrive arbitrarily over time, and processing times are probabilistic. This is more practical job shop circumstances but more complicated to solve it.

1.4 Flow Shop Scheduling Problem

Johnson's Rule (Johnson, 1954) has been the basis of various FSSP heuristics. Palmer (1965) first proposed a heuristic for the FSSP to minimize makespan. FSSP are described by a set of n jobs, where every job has to be processed in the same order on a given number of m machines. Each machine can process only one job at a time. The factors t_{ij},$1 \leq i \leq n$,$1 \leq j \leq m$, indicate The

processing time of job i on machine j [2]. The FSSP is a set of jobs that flow through multiple stages in the similar order as shown in the Figure 3.

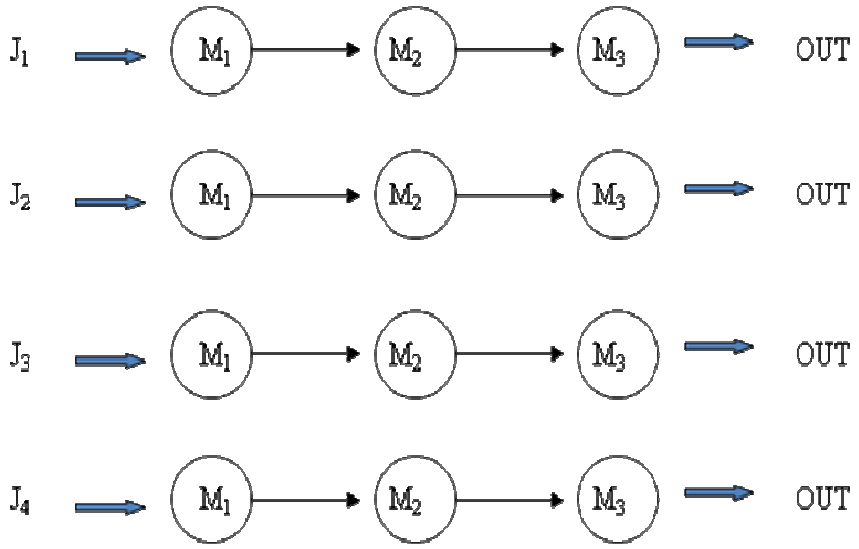

Figure 3: Flow of FSSP

1.5 Open Shop Scheduling Problems

The Open Shop Scheduling Problems can be categorized as n x m, where 'n' is the number of Jobs (J= {j_1, j_2,.., j_n}) can be processed on 'm' number of Machines (M= {m_1, m_2,...,m_m}). Every machine can process at most single operation at a moment and each job can be processed by at most single machine at a time. For every machine the order in which the jobs are processed on the machine (Machine Orders) and for all jobs the order in which this job is processed through the machines (Job Orders) can be selected arbitrarily.

Constraints

- No machine can process more than single operation at the similar time and
- No job can be processed by more than single machine at the similar time.

In this research work, BFO algorithm was hybridized with ACO and a new Customized Bacterial Foraging Optimization (CBFO) algorithm was proposed. Both BFO and CBFO algorithm were applied to Admas, Balas and Zawaxk (ABZ), Carlier (Car), Taillard (TA) and Ravibabu, Narendhar and Divya (RND) randomly created instances. The results obtained by CBFO algorithm is compared and analyzed with BFO and existing algorithms.

2. Related Works

E. Taillard [1989] has proposed a paper about Benchmarks' for Basic Scheduling Problems. In this paper Taillard talked on the subject of 260 scheduling problems whose size is greater than

that of other examples. In this paer he explained about Job Shop, Flow Shop, Open Shop Scheduling Problems. The objective of this paper is to mininmization of makespan [12]

Ashwani Kumar Dhingra has discussed about scheduling problems. He gave a brief explanation about scheduling problems, Significance of Scheduling, Scheduling in a Manufacturing System and Classification of scheduling problems based on requirement generations. Problems up to 200 jobs and 20 machines for instances expanded by Taillard (1993) have been solved and proposed metaheuristics can be tested on various problems [5].

Mahanim Omar, Adam Baharum, Yahya Abu Hasan (2006) have proposed a paper about A Job Shop Scheduling Problem (JSSP) using Genetic Algorithm. Job shop problems are generally known as a NP-Hard problem. In this paper they have formed a preliminary population arbitrarily together with the result achieved by some familiar priority rules such as shortest processing time and longest processing time. This is used to reduce the objective function [25].

David Applegate, William Cook (1991) have proposed a paper about A Computational Study of the JSSP. They tested performance of JSSP with some model instances. MT-10 is a well-known 10 by 10 problem of Muth and Thompson; ABZ5 and ABZ6 are two problems from Admas, Balas and Zawaxk; the problems LA19 and LA20 are problems of Lawrence. They compared their results with best solution [10].

According to Hela Boukef, Mohamed Benrejeb and Pierre Borne [2006-2007] have proposed a new genetic algorithm coding is suggested in this paper to solve flow-shop scheduling problems. To explain the effectiveness of the considered approach with decrease of different costs related to every problem as a scope. Multi-objective optimization is thus, used and its performances confirmed. The standard range of this technique, based on natural variety of mechanism, is the development of robustness and balance among cost and performance [15].

AndreasFink, StefanVoß (2001) have projected a paper about Solving the Continuous Flow Shop Scheduling Problem by Metaheuristics. This problem is used to locate a combination of jobs to be processed consecutively on a number of machines under the constraint that the processing of every job has to be uninterrupted with respect to the purpose of minimizing the entire processing time (flow-time). i.e., once the processing of a job begins, there must not be any waiting times between the processing of any consecutive tasks of this job [2].

According to Samia kouki, Mohamed Jemni, Talel Ladhari (2011) have proposed a paper about Solving the Permutation Flow Shop Problem with Makespan principle using Grids. The optimization of scheduling problems is stand on different criteria to optimize. One of the most significant criteria is to reduce the completion time of the final task on the end machine called makespan. They offered a parallel algorithm for solving the permutation flow shop problem. This is used to minimizing the total makespan of the tasks by using Branch and Bound method to find optimal solutions [37]

According to Peter Brucker, Johann Hurink, Bernd Jurisch and Birgit Gstmann (1995) have proposed a paper about fundamental concepts of branch & bound algorithm. The branch and bound algorithm for the OSSP is based on a disjunctive graph formulation. The problem determined a possible mixture of the machine and job orders which minimizes a certain objective function.

Ching-Fang Liaw (1999) has talk on the subject of the growth and function of a Hybrid Genetic Algorithm (HGA) to the OSSP is based on Tabu Search (TS) into a basic Genetic Algorithm (GA). The local enhancement method enables the HGA algorithm to execute genetic search in excess of the subspace of local optima. Benchmark problems of OSSP are tested by using these algorithms. The results were compared with other algorithms also [7].

Jing Dang, Anthony Brabazon, Michael O"Neill, and David Edition (2008) have discussed a paper regarding Bacterial Foraging Optimization (BFO) algorithm. This technique was implemented to resolve the parameter estimation of an EGARCH-M model.During the lifetime of E.coli bacteria, they undergo different stages such as chemotaxis, reproduction and elimination-dispersal. BFO algorithm is applied to solve various real world problems [19]

Chunguo Wu, Na Zhang, Jingqing Jiang, Jinhui Yang, and Yanchun Liang (2007) described Bacterial Foraging algorithm is a novel evolutionary computation algorithm. This is based on the foraging behaviour of E.coli bacteria living in human intestine. BFO is fundamentally an arbitrary search algorithm. This better algorithm is applied to Job Shop Scheduling Benchmark problems [9].

S. Subramanian and S. Padma (2011) have proposed a paper about the selection behaviour of bacteria leans to eliminate poor foraging strategies and get better successful foraging strategies. The E.coli bacterium has a control system that enables it to look for food and try to keep away from noxious substances. BFO is used to reduce the cost and improves the competence concurrently by using a multi objective based bacterial foraging algorithm [35]

According to James Montgomery, cardc Fayad and Sarja Petrovic have proposed a paper about Solution Representation for Job Shop Scheduling Problems in ACO. The result produces improved explanation more quickly than the usual approach. They created resolutions by creating a permutation of the operations, from which a deterministic algorithm can produce the real schedule [17]

Katie Kinzler (2008), has proposed a dissertation about Mathematical Modeling of Ant Pheromones: Purpose of Optimum pheromone Evaporation Rate and replication of Pheromone Tracking Abilities. There are more varieties of technique used by ant to communicate as well as a variety of reasons for communication. These communications involves stroking, gasping, antenna movements, and streaking of chemicals. These chemicals are known as pheromones. This is the major form of communication used by ants [22]

3. Customized Bacterial Foraging Methodology For JSSP, FSSP & OSSP

The objectives of this research paper are

- To propose and implement Customized Bacterial Foraging Optimization (CBFO) to solve JSSP, FSSP and OSSP.
- CBFO is to find a schedule that reduces the makespan of the jobs.
- To examine the efficiency of CBFO in solving benchmark instances of JSSP, FSSP and OSSP.
- To analyze and compare the performance of the proposed CBFO with BFO in solving JSSP, FSSP and OSSP.

3.1 Customized Bacterial Foraging Optimization (CBFO)

The activity of ant structure is included in tumble part of BFO algorithm, to formulate it as a CBFO. Each ant creates a tour by repeatedly applying a stochastic greedy rule, which is called the state transition rule.

$$s = \begin{cases} \arg max_{u \in J(r)}\{[\tau(r,u)].[\eta(r,u)^{\beta}]\}, & if\ q \le q_0 (exploitation) \\ S, & otherwise\ (biased\ exploitation) \end{cases} \rightarrow \quad (1)$$

(r, u) represents an boundary between point r and u, and $\tau(r, u)$ represents the pheromone on border (r, u). $\eta(r, u)$ is the attraction of border (r, u), which is habitually described as the contrary of the length of edge (r, u). q is a arbitrary number uniformly distributed in [0, 1], q_0 is a user-defined parameter with ($0 \le q_0 \le 1$), β is the parameter controlling the relative importance of the desirability. J (r) is the set of edges available at decision point r. S is a arbitrary variable selected according to the probability distribution given below.

$$P(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)^{\beta}]}{\sum_{u \in J(r)}[\tau(r,u)].[\eta(r,u)^{\beta}]}, & if\ s \in J(r) \\ 0, & otherwise \end{cases} \rightarrow \quad (2)$$

The mixture approach used on top of this is also called 'roulette wheel' selection since its mechanism is an imitation of the process of a roulette wheel [16].

While ant goes for a search it will drop a certain amount of pheromone. It is a continuous progression, but we can regard it as a discrete release by some rules. There are two kinds of pheromone update strategies, called local updating rule and the global updating rule.

Local updating rule

While ant generating its tour, ant will adjust the quantity of pheromone on the passed perimeters by applying the local updating rule.

$$\tau(r,s) \leftarrow (1-\rho).\tau(r,s) + \rho.\tau o \quad \rightarrow \quad (3)$$

Where ρ is the coefficient representing pheromone evaporation (note:$0 < \rho < 1$).

Global updating rule

Once all ants have entered at their target, the amount of pheromone on the boundary is modified again by applying the global updating rule.

$$\tau(r,s) \leftarrow (1-\alpha).\tau(r,s) + \alpha.\Delta\tau(r,s) \quad \rightarrow \quad (4)$$

Where

$$\Delta\tau(r,s) \begin{cases} (L_{gb})^{-1}, & if\ (r,s) \in global-best-tour \\ 0, & otherwise \end{cases} \rightarrow \quad (5)$$

$0<\alpha<1$ is the pheromone decompose parameter, and L_{gb} is the distance of the globally most excellent tour from the starting of the examination. $\Delta\tau(r; s)$ is the pheromone addition on edge (r, s). We can see that only the ant that discovers the global best tour can attain the pheromone increase [15].

In BFO, the objective is to discover the least of $J(\theta), \theta \in R^D$, where we do not have the gradient information $J(\theta)$. Suppose θ is the location of the bacterium and $J(\theta)$ stands for a nutrient profile,

i.e.,J(θ) < 0, J(θ)=0 and J(θ)> 0 stand for the presence of nutrients, a neutral medium and noxious substances correspondingly. The bacterium will try to go towards increasing concentrations of nutrients (i.e. find lower values of J), search for ways out of neutral media and avoid noxious substances (away from positions where J > 0). It equipment a kind of biased random walk.

The mathematical swarming (cell-cell signalling) function can be characterized by:

$$J_{cc}\left(\theta^{i},\theta\right)=\begin{cases}-M\left(\sum_{k-1}^{s}e^{-w_a\|\theta^{i}-\theta^{k}\|^{2}}-\sum_{k-1}^{s}e^{-w_r\|\theta^{i}-\theta^{k}\|^{2}}\right) & With\ swarming\\ 0 & No\ swarming\end{cases}\rightarrow(6)$$

Where $\|.\|$ is the Euclidean norm, W_a and W_r are actions of the width of the attractant and repulsive signals correspondingly, M measures the magnitude of the cell-cell signaling consequence [15].

The above State Transition rule of ant in ACO is included in the tumble. CBFO methodology is implemented with no swarming effect (ie) j_{cc}=0 [19]. Here time is considered as cost. At some point in the lifetime of E-Coli bacteria they undertake different phases such as Chemotactics, Reproduction and Elimination-Dispersal. When compared with ACO and BFO, CBFO attains high level of SHA1PRNG algorithm incase of reproduction, elimination-dispersal.

CBFO Algorithm

for *Elimination-dispersal* **do**
  for *Reproduction* **do**
    for *Chemotaxis* **do**
      for *Bacterium i* **do**
        Tumble: Generate a secure random vector q $\in$ decimal value.
        If q < q_0 **then**
          Generate a secure random vector l $\in$ operation, according to pheromone value ph[job][operation] based on equation 1.
        Else
          Generate a secure random vector l $\in$ operation, according to pheromone value ph[job][operation] based on equation 2.
        end
        Move: Generate a secure random vector l_{new} $\in$ operation.
          Swim:
          if time[job][l] < time[job][l_{new}] **then**
            current_operation = l
          Else
            current_operation = l_{new}
          end
        end
      end
    end
    Sort bacteria in order of ascending time J_{st}. The $S_r = S/2$ bacteria with The peak J value die and other S_r bacteria with the preeminent value split Update value of J and J_{st} consequently.
  end
  Eliminate and disperse the bacteria to arbitrary locations on the optimization domain with probability p_{ed}. Update equivalent J and J_{st}.
End

Note: Parameters are described below in Nomenclature

3.2 Nomenclature

J_{cc}	-	Health of bacterium i
$\omega_{attract}$	-	Width of attractant
$\omega_{repellant}$	-	Width of repellent
J^{i}_{health}	-	Health of bacterium i
L	-	Counter for elimination- dispersal step
P_{ed}	-	Probability of occurrence of elimination-dispersal events
S	-	Population of the E.coli bacteria

4. Implementation Results and Discussion

This paper discusses and compares the result of the implementation of BFO and proposed CBFO algorithm in solving the Benchmark instances of JSSP, FSSP and OSSP.

Admas, Balas and Zawaxk (ABZ) Benchmark problems [30], Ravibabu, Narendhar and Divya (RND) randomly created instance for JSSP, Carlier (car) benchmark problems [30] and RND for FSSP and Taillard benchmark problems and RND for OSSP were solved in this research work. Benchmark instances were taken from Operations Research (OR) Library to test the efficiency of proposed CBFO. The proposed CBFO algorithm gave reasonable solution for most runs for the constant values $\rho=0.1$, $\beta=1.0$, $\alpha=0.1$, $q0=0.8$, $\tau=0.5$. . The result achieved by proposed CBFO algorithm was compared with BFO and existing algorithms. The CBFO algorithm gave a best makespan for most of the problems.

4.1 JSSP Comparison Results for ABZ Instances

The best result for JSSP achieved from proposed CBFO algorithm and BFO algorithm were compared with optimal value of ABZ Instances are shown in Table 1. The variation of CBFO algorithm is due to Time constraint and limited iterations. The proposed CBFO can be improved to achieve best solution by including the swarming technique and also by adjusting constant values used in the algorithms. The Figure 4 shows the graphical representation of Table 1.

Table 1: JSSP Comparison Results for ABZ Instances

INSTANCE	SIZE	BFO	OPTIMAL[9]	CBFO
ABZ 5	10 *10	1323	1234	**1320**
ABZ 5	10 *10	1012	943	**975**
ABZ 5	20 * 15	787	668	**782**
ABZ 5	20 * 15	822	687	**790**
ABZ 5	20 * 15	856	707	**835**

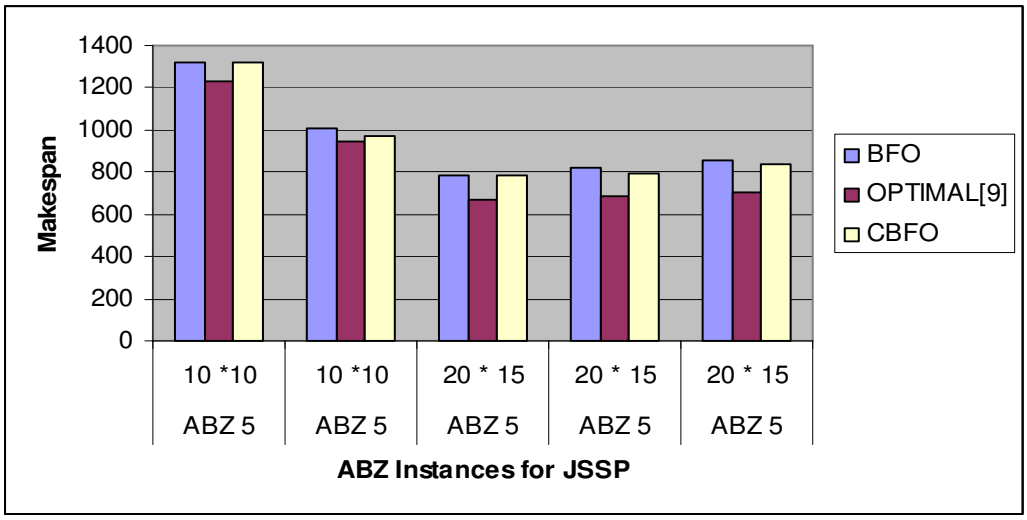

Figure 4: Graphical representation of results for ABZ Instances

4.2 JSSP Comparison Results for RND Instances

The best result obtained from BFO algorithm and optimal values are compared with proposed CBFO algorithm of RND Instances is shown in Table 2. The Figure 5 shows the graphical representation of Table 2.

Table 2: JSSP Comparison Results for RND Instances

INSTANCE	SIZE	BFO	CBFO
RND 10	10*10	740	**709**
RND 20	20*20	1762	**1746**
RND 30	30*30	2652	**2601**
RND 40	40*40	3574	**3457**
RND 50	50*50	4849	**4685**

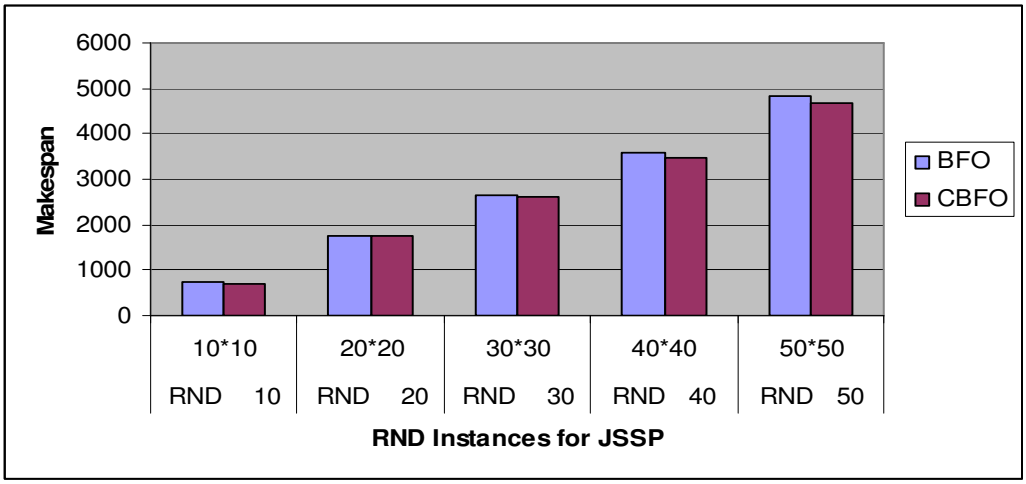

Figure 5: Graphical representation of results for RND Instances

4.3 FSSP Comparison Results for Car Instances

The best result obtained from proposed CBFO algorithm, BFO algorithm were compared with Lower Bound (LB), Upper Bound (UB) [22] of Carlier Instances are shown in Table 3. The Figure 6 shows the graphical representation of Table 3.

Table 3: FSSP Comparison Results for Car Instances

INSTANCE	SIZE	LB	UB	BFO	CBFO
Car 1	11 * 5	7038	7817	7452	**7285**
Car 2	13 * 4	7166	7940	8051	**7640**
Car 3	12 * 5	7312	7779	**7900**	7930
Car 4	14 * 4	8003	8679	8707	**8344**
Car 5	10 * 6	7720	8773	8094	**8365**
Car 6	8 *9	8505	10211	9068	**9656**
Car 7	7 * 7	6590	7043	6868	**6940**
Car 8	8 * 8	8366	9696	8703	**9316**

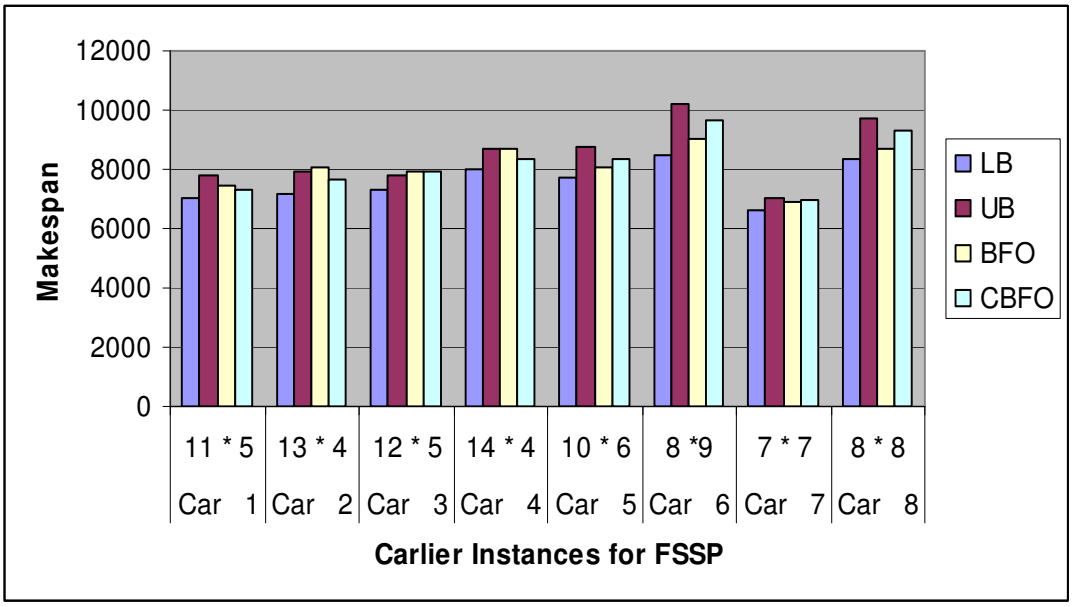

Figure 6: Graphical representation of results for Carlier Instances

4.4 FSSP Comparison Results for RND Instances

The best result obtained from proposed CBFO algorithm is compared with best result obtained from BFO algorithm in solving FSSP for RND instances are shown in Table 4. The Figure 7 shows the graphical representation of Table 4.

Table 4: FSSP Comparison Results for RND Instances

INSTANCE	SIZE	BFO	CBFO
RND 5	5 * 5	463	**461**
RND 6	6 * 6	571	**563**
RND 7	7 * 7	613	**607**
RND 10	10 * 10	1055	**1050**
RND 25	25 * 25	**2920**	2940

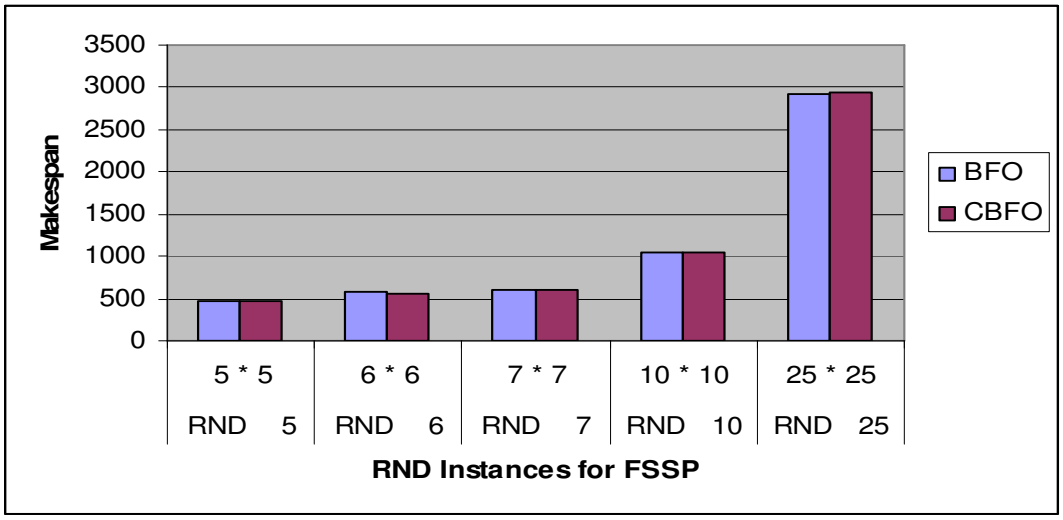

Figure 7: Graphical representation of results for RND Instances

4.5 OSSP Comparison Results for TA Instances

The best solution obtained from proposed CBFO algorithm is compared with best solution obtained from Lower Bound (LB), Upper Bound (UB) of Taillard Instances and BFO algorithm in solving OSSP are shown in Table 5. The Figure 8 shows the graphical representation of Table 5.

Table 5: OSSP Comparison Results for TA Instances

INSTANCE	LB	UB	BFO	CBFO
Ta 4*4	186	193	194	**192**
Ta 4*4	229	236	243	**235**
Ta 5*5	321	328	377	**357**
Ta 5*5	349	353	403	**395**
Ta 7*7	416	419	567	**549**
Ta 7*7	398	400	530	**518**

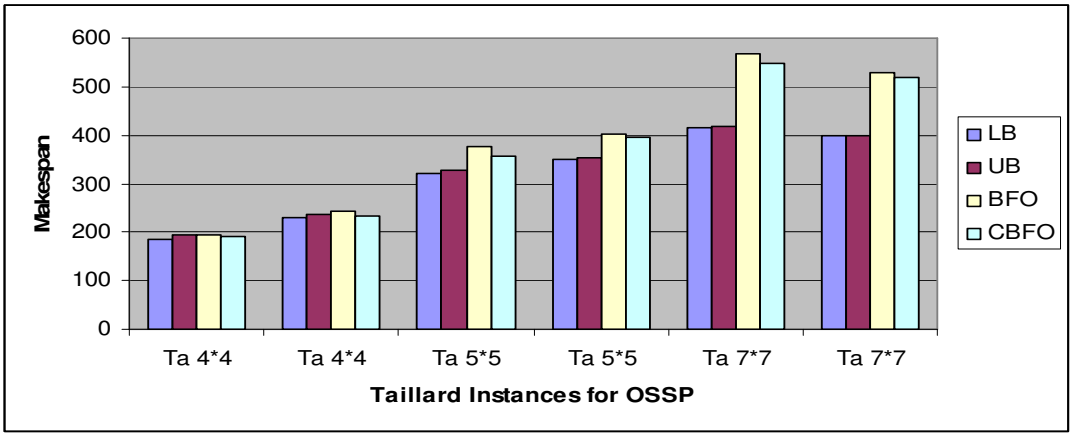

Figure 8: Graphical representation of results for Taillard Instances

4.6 OSSP Comparison Results for RND Instances

BFO algorithm is compared with proposed CBFO algorithm in solving OSSP for RND instances are shown in Table 6. The Figure 9 shows the graphical representation of Table 6.

Table 6: OSSP Comparison Results for RND Instances

INSTANCE	SIZE	ACO	BFO	CBFO
RND 4	4 * 4	212	210	**207**
RND 5	5 * 5	296	290	**287**
RND 7	7 * 7	510	498	**486**
RND 10	10 * 10	834	799	**787**
RND 15	15 * 15	1188	1167	**1121**
RND 20	20 * 20	1628	1446	**1350**

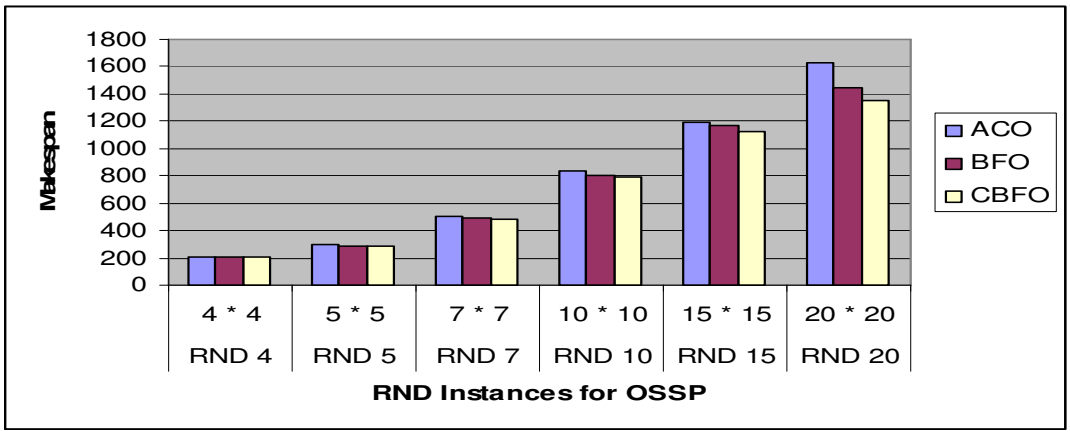

Figure 9: Graphical representation of results for RND Instances

5. Conclusions

In this research work, BFO algorithm is hybridized with ACO and a new technique CBFO was proposed for solving different instances of JSSP, FSSP and OSSP. The proposed CBFO algorithm was investigated through the performance of several runs on well-known test problems of different sizes, which were taken from OR library, which is the primary repository for such problems. The results obtained by the proposed CBFO algorithm for JSSP, FSSP and OSSP can achieve the best and near best solution quality for most of the instances and RND instances.

The implementation of the CBFO algorithm for huge size instances can be done by raising the number of iterations to get best solutions. The proposed CBFO for JSSP, FSSP and OSSP can be improved to achieve best solution by including the swarming technique and also by adjusting constant values used in the algorithms. As a future work, Flexible Job Shop Scheduling, Flexible Flow Shop Scheduling and Flexible Open Shop Scheduling problems can also be solved using proposed CBFO algorithm.

REFERENCES

[1] Ajith Abraham, Arijit Biswas, Sambarta Dasgupta, and Swagatam Das, "Synergy of PSO and Bacterial Foraging Optimization – A Comparative Study on Numerical Benchmarks," Innovations in Hybrid Intelligent Systems, 2008.

[2] AndreasFink, StefanVoß., "Solving the Continuous Flow-Shop Scheduling Problem by Metaheuristics", 2001.

[3] Arnaud Malapert, Hadrien Cambazard, Christelle Guéret, Narendra Jussien, André Langevin, and Louis-Martin Rousseau, "An Optimal Constraint Programming Approach to the Open-Shop Problem." CIRRELT – 2009-25, June 2009.

[4] Ashwani Dhingra and Pankaj Chandna., "Hybrid Genetic Algorithm for Multicriteria Scheduling with Sequence Dependent Set up Time".

[5] Ashwani Kumar Dhingra., "Multi-Objective Flow Shop Scheduling using Metaheurustics", 2006.

[6] C. Blum, "Beam-ACO-Hybridizing ant colony optimization with beam search: an application to open shop scheduling," Computers and Operations Research, No 32; pp. 1565-1591, 2005.

[7] Ching –Fang Liaw, " A hybrid genetic algorithm for open shop scheduling problem", European Journal of Operational Research, Volume 124, Issue 1, pp. 28-42, July 2000

[8] Christelle Guéret, Christian Prins, Marc Sevaux., "Applications of optimization with Xpress-MP",2000.

[9] Chunguo Wu, Na Zhang, Jingqing Jiang, Jinhui Yang, and Yanchun Liang., "Improved Bacterial Foraging Algorithms and their Applications to Job Shop Scheduling Problems", 2007.

[10] David Applegate, William Cook., "A Computational Study of the Job-Shop Scheduling Problem", 1991.

[11] Divya. P, Amudha. T, Narendhar. S "An Improved Bio-Inspired Methodology to solve flow shop scheduling problems", 2012

[12] E. Taillard, "BenchMarks For Basic Scheduling Problems," European Journal of Operations Research, 64, pp. 278-285, 1993.

[13] H. Van Dyke Parunak., "Go to the Ant" Engineering Principles from Natural Multi-Agent Systems, 1997.

[14] Hanning Chen, Yunlong Zhu, and Kunyuan Hu ., "Cooperative Bacterial Foraging Optimization," Hindawi Publishing Corporation, Discrete Dynamics in Nature and Society, Article ID 815247, 17 page, Volume 2009.

[15] Hela Boukef, Mohamed Benrejeb and Pierre Borne., "A Proposed Genetic Algorithm Coding for Flow-Shop Scheduling Problems", 2006-2007.

[16] Ilkyeong. Moon, Jieom. Lee., "Genetic Algorithm Application to the Job Shop scheduling Problem with Alternative Routings", 2000

[17] James Montgomery, cardc Fayad and Sarja Petrovic., "Solution Representation for Job Shop Scheduling Problems in Ant Colony Optimization".

[18] Jason Brownlee., "Clever Algorithms: Nature-Inspired Programming Recipes," First Edition, January 2011.

[19] Jing Dang, Anthony Brabazon, Michael O"Neill, and David Edition., "Option Model Calibration using a Bacterial Foraging Optimization Algorithm", LNCS 4974, 2008.

[20] Jun Zhang, Xiaomin Hu, X.Tan, J.H Zhong and Q. Huang., "Implementation of an Ant Colony Optimization Technique for Job Shop Scheduling Problem," Transactions of the Institute of Measurement and Control 28, pp. 93_ 108, 2006.

[21] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," IEEEControl Systems Magazine, vol. 22, pp. 52–67, 2002.

[22] Katie Kinzler., "Mathematical Modeling of Ant Pheromones: Determination of Optimum pheromone Evaporation Rate and Simulation of Pheromone Tracking Abilities," BBSI Program, Summer, 2008.

[23] Kevin M. Passino., "Bacterial Foraging for Optimization," International Journal of Swarm Intelligence Research, 1(1), pp. 1-16, January – March, 2010.

[24] M. Dorigo, L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," Evolutionary Computation, IEEE Transactions on 1(1), pp. 53–66, 2002.

[25] Mahanim Omar, Adam Baharum, Yahya Abu Hasan., A Job-Shop Scheduling Problem (JSSP) using Genetic Algorithm, 2006.

[26] Marcelo Seido Nagano., A Constructive Genetic Algorithm fo r Permutation Flowshop Scheduling.

[27] Murugesan, S.ThamaraiSelvi., P.Alphonse Rajendran and V.S.SampathKumar., Identication of a Rank Minimal Optimal Sequence for Open Shop Scheduling Problems, 2003.

[28] Narendhar. S and Amudha. T "A Hybrid Bacterial Foraging Algorithm For Solving Job Shop Scheduling Problems", 2012.

[29] O. Seraj and R. Tavakkoli-Moghaddam., "A Tabu Search Method For A New Bi-Objective Open Shop Scheduling Problem By A Fuzzy Multi-Objective Decision Making Approach," IJE Transactions B: Applications Vol. 22, No. 3, October 2009.

[30] Orhan ENGİ N, Alper DÖYEN ., "A New Approach To Solv E Flowshop Scheduling Problems By Artificial Immune Systems",2007.

[31] P. Brucker, J. Hurink, B. Jirisch, B. Wostmann, "Branch and Bound Algorithm for the Open Shop Problem: Discrete Applied Mathematics," Volume 76, pp. 43-59, 1997.

[32] Ravibabu. V, Amudha. T , "An ant inspired Bacterial Foraging Methodology to solve Open Shop Scheduling Problem", 2012

[33] R.Murugesan, S.ThamaraiSelvi., P.Alphonse Rajendran and S. SampathKumar., "Identication of a Rank Minimal Optimal Sequence for Open Shop Scheduling Problems, Information and Management Sciences," Volume 14, Number 1, pp. 37-55, 2003.

[34] Rutger Claes and Tom Holvoet., "Cooperative Ant Colony Optimization in Traffic Route Calculations," IWT - SBO project „MASE" (project no. 060823), 2011.

[35] S. Subramanian and S. Padma., "Bacterial Foraging Algorithm Based Multiobjective Optimal Design of single phase Transformer," Journal of Computer Science And Engineering, Volume 6, Issue 2, April 2011.

[36] Sambarta Dasgupta, Swagatam Das, Ajith Abraham, Senior Member, IEEE, and Arijit Biswas., "Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis," IEEE Transactions on Evolutionary Computation, vol. 13, no. 4, August 2009.

[37] Samia kouki, Mohamed Jemni and Talel Ladhari., "Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids", 2011.

[38] Seda HEZER, Yakup KARA., "Solving Vehicle Routing Problem with Simultaneous Delivery and Pick-up using Bacterial Foraging Optimization Algorithm".

[39] Teofilo Gonzalez and Sartaj Sahni., "Open Shop Scheduling to Minimize Finish Time," Journal of Association for Computing Machinery. 23(4), pp. 665-679, 1976.

[40] Vittorio Maniezzo, Luca Maria Gambardella and Fabio de Luigi., "Ant Colony Optimization," Addison-Wesley, pp. 101-117, 2004. [27] J. E.

[41] http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Authors

Ms. P. Divya received her B. Sc Degree in Computer Science, Masters Degree (MCA) in Computer Applications in 2009 and 2012 respectively from Bharathiar University, Coimbatore, Tamil Nadu, India. Her area of interest is Bio-Inspired computing

Mr. S. Narendhar received his B. Sc Degree in Computer Technology from Anna University, Coimbatore, India in the year 2009 and Masters Degree (MCA) in Computer Applications from Bharathiar University, Coimbatore, India in the year 2012. His area of interest includes Agent based computing and Bio-inspired computing. He has attended National Conferences.

Mr. V. Ravibabu received his B.Sc Degree in Computer Science and MCA Degree in Computer Applications in 2009 and 2012 respectively, from Bharathiar University, Coimbatore, India. His area of interest includes Agent based computing and Bio-inspired computing. He has attended National Conferences. He is a member of International Association of Engineers.