Global Journal of Computer Science and Technology
Hardware & Computation
Volume 12 Issue 10 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Hybrid Bacterial Swarming Methodology for Job Shop Scheduling Environment

By Shivakumar B L & Amudha T
SNR sons College, Coimbatore, India

Abstract - Optimized utilization of resources is the need of the hour in any manufacturing system. A properly planned schedule is often required to facilitate optimization. This makes scheduling a significant phase in any manufacturing scenario. The Job Shop Scheduling Problem is an operation sequencing problem on multiple machines with some operation and machine precedence constraints, aimed to find the best sequence of operations on each machine in order to optimize a set of objectives. Bacterial Foraging algorithm is a relatively new biologically inspired optimization technique proposed based on the foraging behaviour of E.coli bacteria. Harmony Search is a phenomenon mimicking algorithm devised by the improvisation process of musicians. In this research paper, Harmony Search is hybridized with bacterial foraging to improve its scheduling strategies. A proposed Harmony Bacterial Swarming Algorithm is developed and tested with benchmark Job Shop instances. Computational results have clearly shown the competence of our method in obtaining the best schedule.

Keywords : *Harmony Search, BFO, Scheduling, Bioinspired systems, optimization.*

GJCST-A Classification: *D.4.1*

Strictly as per the compliance and regulations of:

A Hybrid Bacterial Swarming Methodology for Job Shop Scheduling Environment

Shivakumar B L [α] & Amudha T [σ]

Abstract - Optimized utilization of resources is the need of the hour in any manufacturing system. A properly planned schedule is often required to facilitate optimization. This makes scheduling a significant phase in any manufacturing scenario. The Job Shop Scheduling Problem is an operation sequencing problem on multiple machines with some operation and machine precedence constraints, aimed to find the best sequence of operations on each machine in order to optimize a set of objectives. Bacterial Foraging algorithm is a relatively new biologically inspired optimization technique proposed based on the foraging behaviour of E.coli bacteria. Harmony Search is a phenomenon mimicking algorithm devised by the improvisation process of musicians. In this research paper, Harmony Search is hybridized with bacterial foraging to improve its scheduling strategies. A proposed Harmony Bacterial Swarming Algorithm is developed and tested with benchmark Job Shop instances. Computational results have clearly shown the competence of our method in obtaining the best schedule.

Keywords : Harmony Search, BFO, Scheduling, Bio-inspired systems, optimization.

I. Introduction

Nature Inspired Computing (NIC) is an upcoming area of research that aims at developing innovative computing techniques by observing how nature behaves in various situations to solve complex problems [2]. Based on the observation from nature a problem solving strategy can be formulated. The strategy can be used to design an initial model and remodeling it until a near perfect working model is obtained. The resulting model may also find certain new and unknown mechanisms. Principles such as survival of the fittest and law of jungle are used to develop the nature-inspired approaches [3]. NIC techniques are highly adaptable that they can be applied to wide range of problems and can be dealt with unseen data and even incomplete data. capable of learning, so robust that can handle. They have decentralized control of computational activities. Biological Inspired Computing (BIC) is a subdivision of Nature inspired computing (NIC). Bio- inspired algorithms are characterized byalgorithmic operators mimicking computationally useful aspects of various biological phenomena. Metaheuristic are algorithmic templates used to specify problem-independent optimization strategies, which can be instantiated in order to define problem-specific heuristics [24] [25].

In computer science, metaheuristics designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Metaheuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics do not guarantee an optimal solution in most cases. Many metaheuristics implement some form of stochastic optimization. Metaheuristics have been most generally applied to problems classified as NP-Hard or NP-Complete by the theory of computational complexity. Some of the most successful metaheuristic [7] conceived in the last few years are Tabu Search, Simulated Annealing, Genetic Algorithms and Memetic Algorithms, Ant Colony Optimization (ACO) and Bacterial Foraging Optimization (BFO), Bee algorithms and Harmony Search (HS). They are population-based methods that make use of the global behavior that emerges from the local interaction of individuals with one another and with their environment.

Bacterial Foraging optimization algorithm is a novel biologically inspired computing technique proposed based on the foraging behavior of Escherichia coli (E. coli) bacteria living in human intestine [4]. The BFO Algorithm belongs tofield of Bacteria Optimization Algorithms and Swarm Optimization, and more broadly to thefields of Computational Intelligence and Metaheuristics. It is related to other Bacteria Optimization Algorithms such as the Bacteria Chemotaxis Algorithm and other Swarm Intelligence algorithms such as ACO and PSO. In specific, the BFO Algorithm is inspired by the chemotaxis behavior of bacteria that will perceive chemical gradients in the environment (such as nutrients) and move toward or away from specific signals.

Harmony Search is a phenomenon mimicking algorithm by the improvisation process of musicians. In the HS algorithm, each musician (= decision variable) plays (= generates) a note (= a value) for finding a best harmony (= global optimum) altogether [26]. HS algorithm was recently developed in an analogy with music improvisation process where music players improvise the pitches of their instruments to obtain

Authors α : Professor & Head, Dept. of Computer Applications, SNR sons College, Coimbatore, India. E-mail : blshiva@yahoo.com
Authors σ : Assistant Professor, Dept. of Computer Applications, Bharathiar University, Coimbatore, India.
E-mail : amudha.swamynathan@gmail.com

better harmony. The working principle of HS algorithm is very different from classical optimization techniques. HS algorithm uses a random search, which is based on the harmony memory considering rate and the pitch adjusting rate. Compared to earlier meta-heuristic optimization algorithms, the HS algorithm imposes fewer mathematical requirements and can be easily adopted for various types of engineering optimization problems It has been successfully applied to various benchmark and real-world problems [1] [8] [13] [14].

ACO belongs to the class of metaheuristics, which are approximate algorithms used to obtain good enough solutions to hard combinatorial optimization problems in a reasonable amount of computation time [6] [10]. The inspiring source of ACO is the foraging behavior of real ants. The ant deposits a chemical pheromone trail on the ground when it forages. The quantity of pheromone deposited depends upon the quantity and quality of the food that will guide other ants to the food source [12]. The indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food sources.

Scheduling problem is one of the most important problems in the field of combinatorial optimization [CO], and it finds its application in various engineering and manufacturing industries. It is generally NP-hard CO problem. The scheduling problem is a problem of deciding which job in which order to be done with which machine, when several jobs are processed with several machines. The Job Shop Scheduling Problem (JSSP) is one of the most difficult problems of these types. It is an operation sequencing problem on multiple machines subject to certain precedence constraints among the operations [5] [7] [28]. The JSSP has several machines of which each machine has each function, and each job is processed according to a specific order. The JSSP is to find the best sequence of operations on each machine in order to minimize or maximize a specific objective or a set of objectives. JSSP belongs to the class of NP-Complete problems and hence various heuristic approaches have been developed to solve these problems.

In this paper, BFO is improved by modifying the swarming behavior of the bacteria. Harmony search is used for improvising the search strategy of the bacteria which has remarkably resulted in a much better schedule. BFO and improved BFO using harmony search method, named as Harmony Bacterial Swarming Algorithm [HBSA] are applied to solve various benchmark instances of JSSP. JSSP benchmark instances were taken from the OR-Library. These instances are worth solving as they are considered as the most significant, classical and complex CO problems known to be NP-hard. The effectiveness of hybrid Bacterial swarming methodology is analyzed by comparing its performance with BFO and ACO. Experimental results on various benchmark instances have clearly shown the competence of the nature-inspired methods in solving complex combinatorial optimization problems and the improvement in optimal solutions when BFO was hybridized with HS.

II. Related Work

Zong Woo Geem et al., (2005) applied Harmony search (HS) to a TSP-like NP-hard Generalized Orienteering Problem (GOP) which is to find the utmost route under the total distance limit while satisfying multiple goals. The results of HS showed that the algorithm could find good solutions when compared to those of artificial neural network.

Sukayapong Ngonkham and Panhathai Buasri (2008) presented harmony search algorithm (HS) to solve economic dispatch (ED) problem in the power system integrating wind energy conversion system (WECS). Three optimization techniques, genetic algorithm (GA), interior point methods (ITP) and HS were applied to solve ED when system connecting and disconnecting to WECS. The results showed that HS can give the better solution than the others. In the comparison with GA, HS and ITP, HS had better than GA 3.2% and better than ITP 1.6%. Moreover total cost reduction when the system connected to WECS computed by HS was reduced to 8% per day.

Quan-Ke Pan et al., (2011) proposed a local-best harmony search (HS) algorithm with dynamic sub-harmony memories (HM), namely DLHS algorithm to minimize the total weighted earliness and tardiness penalties for a lot-streaming flow shop scheduling problem with equal-size sub-lots. Computational experiments and comparisons showed that the proposed DLHS algorithm generated better or competitive results than the existing hybrid genetic algorithm (HGA) and hybrid discrete particle swarm optimization (HDPSO) for the lot-streaming flow shop scheduling problem with total weighted earliness and tardiness criterion.

Mohammed Azmi Al-Betar and Ahamad Tajudin Khader (2010) applied a HS and a modified harmony search algorithm to university course timetabling against standard benchmarks. The results showed that the proposed methods were capable of providing viable solutions in comparison to previous works. The results of modified harmony search algorithm (MHSA) basically outperformed those obtained by basic harmony search algorithm (HSA) significantly. However, the computational time needed for MHSA is longer.

Christian Blum (2005) presented the Ant Colony Optimization algorithm introduction and its recent trends. In their paper, they explained how ant behavior exploited to search the approximate solutions for discrete and continuous optimization problems and to

important problem in telecommunication, such as routing &load balancing problems.

Jun Zhang, Xiaomin Hu, X.Tan, J.H Zhong and Q. Huang (2006) presented an investigation into the use of an Ant Colony Optimization (ACO) to optimize the JSSP. ACO is extensively used to solve NP-Hard Combinatorial Optimization problems. Its original model is based on the foraging behaviour of real ants who find an approximately shortest way to the food by detecting the density of pheromone deposited on the route. The main characteristics of ACO are positive feedback, distributed computation, robustness and the use of a constructive greedy heuristic.

James Montgomery, Cardc Fayad and Sarja Petrovic (2005) have discussed ACO for Job Shop Scheduling Problems and generated solutions by constructing a permutation of the operations, from which a deterministic algorithm can generate the actual schedule. They proposed a paper about Solution Representation for Job Shop Scheduling Problems in ACO. The result produces better solutions more quickly than the traditional approach.

W. J. Tang et al., (2006) studied a bacterial foraging algorithm (BFA) aiming for optimization in dynamic environments, called DBFA. A test bed proposed previously was adopted to evaluate the performance of DBFA. The simulation studies offered a range of changes in a dynamic environment. The simulation results showed that DBFA could adapt to various environmental changes which occurred in different probabilities, with both satisfactory accuracy and stability, in comparison with a recent work on bacterial foraging.

Sambarta Dasgupta et al., (2008) introduced a micro-bacterial foraging optimization algorithm, which evolved with a very small population compared to its classical version. In this modified bacterial foraging algorithm, the best bacterium was kept unaltered, whereas the other population members were reinitialized. This new small population μ-BFOA was tested over a number of numerical benchmark problems for high dimensions and found to outperform the normal bacterial foraging with a larger population as well as with a smaller population.

Wei Liu et al., (2011) presented a novel optimal scheduling method for Radio Frequency Identification (RFID) network using a Self-adaptive Bacterial Foraging Optimization (SABFO) Algorithm. The SABFO, which was based on the recently developed Bacterial Foraging Optimization (BFO) technique, could adjust the run-length unit parameter dynamically during evolution to balance the exploration/exploitation tradeoff. Simulation on an RFID reader network architecture was given to illustrate the effectiveness of the proposed SABFO based scheduling method. The simulation results, which compared to GA, PSO, and BFO, show that the SABFO obtained superior solutions than all the other methods.

According to S. Subramanian and S. Padma (2011), the selection behaviour of bacteria tends to eliminate poor foraging strategies and improve successful foraging strategies. The E.coli bacterium has a control system that enables it to search for food and try to avoid noxious substances. BFO was used to minimize cost and improve the efficiency simultaneously by using a multi objective based bacterial foraging algorithm.

E. Taillard [1989] has proposed a paper about Benchmarks' for Basic Scheduling Problems. In this paper he discussed about 260 scheduling problems whose size is greater than that of the rare examples published. Such sizes correspond to real dimensions of industrial problems. In this paper he solved the permutation Flow Shop, Job Shop and Open Shop Scheduling Problems.

III. Jssp Solution Representation Sing Hybrid Bacterial Swarming Algorithm [hbsa]

a) Bacterial Foraging Algorithm [BFA]

Bacterial foraging was proposed by Kevin M Passino in the year 2002 as a tool for distributed optimization and control which mimics the foraging behavior of E. coli bacteria. E. coli bacteria exist in intestines of most animals on the earth. E. coli bacterium has a control system, which directs its behaviors in food foraging. The foraging process consists of a series moves towards food sources [11]. The control system is in charge of evaluating changes from one state to the other states to provide reference information for E. coli bacterium's next state change. A change between two states is called as a move and the states include advancing direction and step length. E. coli bacteria gradually approach their food sources under the influence of its control system.

Biological studies have shown that the foraging process includes four steps: (1) search for a possible food region, (2) decide to whether or not enter into the possible food region, (3) perform a careful search if it enters into a new region, (4) decide to either keep stay in the current region or emigrate into a new and more ideal region, after they consume some food in the current region. In general, if the bacteria are trapped into a region in deficiency of food, they might draw a conclusion based on past experience that other regions must be in abundance of food. Due to this conclusion, bacteria would change their states [17][18][19]. Hence, each decision of state change is made under the physiological and environmental constraints with the final aim to maximize the obtained energy in unit time [81].

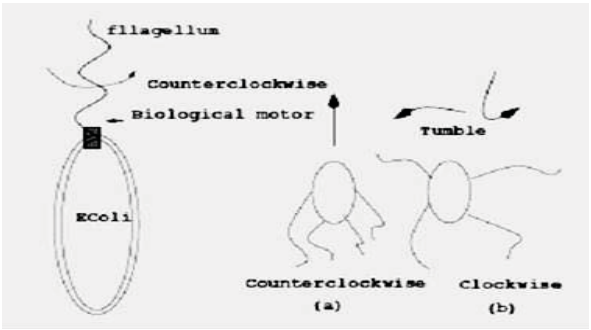

Fig. 1: Bacteria Swarming and Tumbling

The bacterial foraging system consists of three principal mechanisms, namely chemotaxis, reproduction and elimination-dispersal.

Chemotaxis : Biologically an E.coli bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. Chemotaxis simulates the movement of an E.coli cell through swimming and tumbling via flagella. Depending upon the rotation of Flagella in each bacterium, it decides whether it should move in a predefined direction (swimming) or altogether in different directions (tumbling), in the entire lifetime. To represent a tumble, a unit length random direction, say ϕ (j), is generated; this will be used to define the direction of movement after a tumble. In particular

$$\theta^i (j+1, k, 1) = \theta^i (j, k, 1) + C(i)\phi(j) \quad (1)$$

where θi (j, k, l) represents the i-th bacterium at j-th chemotactic, k-th reproductive and i-th elimination and dispersal step. C (i) is the size of the step taken in the random direction specified by the tumble (run length unit).

Reproduction : The least healthy bacteria die and the other healthiest bacteria each split into two bacteria, which are placed in the same location. This makes the population of bacteria constant.

Elimination and Dispersal: It is possible that in the local environment the live of a population of bacteria changes either gradually (e.g., via consumption of nutrients) or suddenly due to some other influence. Events can occur such that all the bacteria in a region are killed or a group is dispersed into a new part of the environment. They have the effect of possibly destroying the chemotactic progress, but they also have the effect of assisting in chemotaxis, since dispersal may place bacteria near good food sources. From a broad perspective, elimination and dispersal are parts of the population- level long-distance motile behavior [58].

Pseudo code of Bacterial Foraging Algorithm

for *Elimination-dispersal loop* **do**
for *Reproduction loop* **do**
for *Chemotaxis loop* **do**
for *Bacterium i* **do**
Tumble: Generate a random vector $\varphi \in R^D$ *in* rand direction
Move: Let $\theta^{new} = \theta i + c\varphi$, compute J^{new}.
Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
Swim: Let m=0
while $m < N_s$ **do**
let m=m+1
if $J^{new}_{sw} < J_{sw}$ **then**
Let $\theta^i = \theta^{new}$, compute Ji and J^i_{sw}
Let $\theta^{new} = \theta^i + c\varphi$, *compute* $J(\theta^{new})$.
Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
Else
Let $m = N_s$
end
end
end
end
Sort bacteria ascending cost J_{sw}.
$S_r = S/2$ bacteria with the highest J value die & other S_r bacteria with the best value split
Update value of J and J_{sw} accordingly.
end
Eliminate and disperse the bacteria to random locations with probability p_{ed}.
Update corresponding J and J_{sw}.
End

Here C(i) is the Step size, i is the Bacterium number, j is the Counter for chemotactic step, J(i, j, k, l) is the Cost at the location of i^{th} bacterium, J_{cc} is the Swarm attractant cost , J^{i}_{health} is the Health of bacterium i, k is the Counter for reproduction step, l is the Counter for elimination- dispersal step, m is the Counter for swimming locomotion, N_c is the Maximum number chemotactic steps, N_{ed} is the Number of elimination-dispersal events, N_{re} is the Maximum number of reproduction steps, Ns is the Maximum number of swims, P is the Dimension of the optimization problem, P_{ed} is the Probability of occurrence of elimination-dispersal events, s is the Population of the E. coli bacteria, $\theta^i(j, k, l)$ is the Location of the i^{th} bacterium at j^{th} chemotactic step, k^{th} reproduction step, and l the elimination-dispersal step, $\omega_{attract}$ is the Width of attractant, $\omega_{repellant}$ is the Width of repellent, $h_{repellent}$ is the Height of repellent, $d_{attract}$ is the Depth of attractant.

The mathematical swarming (cell-cell signaling) function can be represented by:

$$J_{cc}(\theta^i,\theta)=\begin{cases}-M\left(\sum_{k=1}^{S}e^{-W_a\|\theta^i-\theta^k\|^2}-\sum_{k=1}^{S}e^{-W_r\|\theta^i-\theta^k\|^2}\right) & \text{With swarming}\\ 0 & \text{No swarming}\end{cases} \quad (2)$$

Where $\|.\|$ is the Euclidean norm, W_a and W_r are measures of the width of the attractant and repellent signals respectively; M measures the magnitude of the cell-cell signaling effect. These parameter values are chosen according to the nutrient profile used. During the lifetime of E. coli bacteria, they undergo different stages such as chemotactics, reproduction and elimination-dispersal.

b) Harmony Search [HS]

Harmony Search was first proposed by Zong Woo Geem et al. in 2001. This meta-heuristic algorithm was inspired by musical process of searching for a perfect state of harmony [23][27]. The harmony in music is analogous to the optimization solution vector, and the musician's improvisations are analogous to local and global search schemes in optimization techniques. In the HS algorithm, musical performances seek a perfect state of harmony determined by aesthetic estimation, as the optimization algorithms seek a best state (i.e., global optimum) determined by objective function value. It has been successfully applied to various optimization problems in computation and engineering fields including economic dispatch of electrical energy, multicast routing, clustering, optimum design, traveling salesman problem, parameter optimization of river flood model, design of pipeline network, and design of truss structures [15].

A musically pleasing harmony can be found based on three musical rules: (i) by playing a note from harmony memory (HM); (ii) by playing a note which is closer to another note stored in HM; and (iii) by playing an arbitrary note from the entire note range. Combination of these rules allows finding a musically pleasing harmony (best state). Adaptation of these rules to the optimization problems is as follows: (i) generate a new solution vector from HM (memory consideration); (ii) replace a decision variable with a new one which is close to the current one (pitch adjusting); and (iii) generate a solution vector from the possible random range (random selection). Combined utilization of these rules allows identification of the optimal or near optimal solutions [45].

Pseudo code of Harmony Search Algorithm

STEP1. Initialize the problem and HS parameters
Input data. *The data instance of the optimization problem and the HS parameters (HMCR, PAR, NI, HMS)*
STEP2. Initialize the harmony memory
Construct the vectors of the harmony memory
$HM = \{x^1, x^2, \ldots, x^{HMS}\}$
Recognize the worst vector in HM
$x^{worst} \in \{x^1, x^2, \ldots, x^{HMS}\}$
STEP3. Improvise a new harmony
$x' = \varphi$ // *new harmony vector*
for i = 1, . . . , N do // *N is the no. of decision variables.*
if (U(0, 1) ≤ HMCR) then // *U is uniform random no.gen*
begin
$x'_i \in \{x^1_i, x^2_i, \ldots, x^{HMS}_i\}$ {* *memory consideration* *}
if (U(0, 1) ≤ PAR) then
$x'_i = v_{i,k\pm m}$ // $x'_i = v_{i,k}$ {**pitch adjustment* *}
end
else
$x'_i \in X_i$ {* *random consideration* *}
end if
end for
STEP4. Update the harmony memory (HM)
if $(f(x^{new}) < f(x^{worst}))$ *then*
Include x^{new} *to the HM.*
Exclude x^{worst} *from HM.*
STEP5. Check the stop criterion
while (not termination criterion is specified by NI)
Repeat ***STEP3*** *and* ***STEP4***

where HMS is the Harmony Memory Size, HMCR is the Harmony Memory Considering Rate, PAR is the Pitch Adjusting Rate, NI is the no. of iterations.

After initializing the problem parameters, the Harmony Memory (HM) matrix is filled with as many randomly generated solution vectors as the size of the HM (HMS). A new harmony vector, $x' = (x_1', x_2' \ldots x_N')$ can be generated by following HM consideration, Pitch adjustment or totally random generation. The value of the decision variables for the new vector can be chosen from values stored in HM ($x_1^1 \sim x^{HMS}{}_1$). This method also permits to choose totally random values. HMCR parameter, which varies between 0 and 1, sets the rate whether a value stored in HM is chosen or a random value is chosen, as follows:

$$x'_i \leftarrow \begin{cases} x'_i \in \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} & \text{w.p. } HMCR \\ x'_i \in X_i & \text{w.p. } (1\text{-}HMCR) \end{cases} \quad (3)$$

The HMCR is the rate of choosing one value from historical values stored in HM while (1-HMCR) is the rate of randomly choosing one value from the possible value range. On choosing a New Harmony vector $x' = (x_1', x_2' \ldots x_N')$, pitch-adjusting decision is examined for each component of the new vector.

$$x_i' \leftarrow \begin{cases} \text{Adjusting Pitch} & \text{w.p.} \quad PAR \\ \text{Doing Nothing} & \text{w.p.} \quad (1-PAR) \end{cases} \tag{4}$$

In the pitch adjusting process, a value moves to its neighboring value with probability of PAR, or just stays in its original value with probability (1-PAR). The HMCR and PAR parameters in Harmony Search help the algorithm find globally and locally improved solution, respectively.

c) Proposed Hybrid Method-HBSA

The problem solving efficiency of metaheuristics has paved way for the development of many hybrid approaches which combine the best features of more than one metaheuristic to improve the algorithmic performance [9][16]. In our proposed hybrid method HBSA, the best features of harmony search algorithm were incorporated into BFO to obtain improved swarming behavior. In BFO, chemotactic event is the primary and most significant component. It corresponds to the direction selection scheme which is the central step employed by a living creature to search food and in charge of the decisions that whether or not enter into a new region, how long does the individual stay in the current region, which direction should be selected in the next move. These decisions make chemotactic event highly influential in algorithm convergence [22]. The chemotactic event includes swarming as the key phenomena to decide the location in the search space where the bacteria moves next.

Swarming: An interesting group behavior has been observed for several motile species of bacteria including E.coli and S. typhimurium, where stable spatio-temporal patterns (swarms) are formed in semisolid nutrient medium. A group of E.coli cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo effecter[20]. The cells when stimulated by a high level of succinate, release an attractant aspirate, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. The mathematical representation for swarming can be represented by

$$\begin{aligned} J_{cc}(\theta, P(j,k,1)) &= \sum_{i=1}^{s} J_{cc}^{i}(\theta, \theta^{i}(j,k,1)) \\ &= \sum_{i=1}^{s}\left[-d_{attract}\exp\left[-W_{attract}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right]\right] + \\ &\sum_{i=1}^{s}\left[h_{repelent}\exp\left[-W_{repelent}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right]\right] \end{aligned} \tag{5}$$

where *Jcc*(θ, *P*(*j*, *k*, *l*)) is the cost function value to be added to the actual cost function to be minimized to present a time varying cost function. "*S*" is the total number of bacteria and "*p*" the number of parameters to be optimized which are present in each bacterium. $d_{attract}$, $w_{attract}$, $h_{repellent}$, $w_{repellent}$ are different coefficients that are to be chosen properly.

To implement swarming, the improvisation technique of the HS algorithm is used in our proposed method. This procedure uses the PAR parameter to set the rate of pitch adjustment as follows:

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix} \begin{matrix} \Rightarrow f(x^1) \\ \Rightarrow f(x^2) \\ \Rightarrow \vdots \\ \Rightarrow f(x^{HMS-1}) \\ \Rightarrow f(x^{HMS}) \end{matrix} \tag{6}$$

Since HS does not require any initial values for the decision variables, the procedure to decide the swarming nature becomes highly flexible. In improvisation process, a value moves to its neighboring value with probability of pitch adjusting rate, or just stays in its original value with probability. The HMCR and PAR parameters in Harmony Search help the algorithm find globally and locally improved solution, respectively. Also, HS algorithm uses a stochastic random search that is based on the harmony memory considering rate and the pitch adjusting rate so that derivative information becomes irrelevant. Harmony search algorithm can deal with discrete variable problems as well as continuous variable problem and hence, it can also be applied for parameter optimization to identify the suitable set of parameter values for the optimization algorithms.

The job shop scheduling problem (JSSP) is the most popular scheduling model in practice and it has attracted many researchers due to its wide applicability and inherent difficulty. A Job Shop Scheduling Problem (JSSP) refers to the static problem where optimal schedules are searched for a given set of jobs. It is generally NP-hard [10][15][21]. In the real world, a JSSP becomes dynamic when jobs arrive continuously and it thus has an additional complexity. A JSP may be formulated as follows: given an $n \times m$ static JSP, in which n jobs must be processed exactly once on each of *m* machines, the set of *n* jobs can be defined as J= $\{J_1, \ldots\ldots, J_n\}$, while the set of *m* machines is M= $\{M_1, \ldots, M_m\}$. Each job is routed through the *m* machines in a pre-defined order, which is also known as operation precedence constraints. The processing of a job on one machine is called an operation, and the processing of job *i* on machine *j* is denoted by u_{ij}. So the set of operations can be defined as $O = \{ u_{ij} \mid i \in [1, n], j \in [1, m]\}$, in which *n* denotes the number of jobs and *m* denotes the number of machines. Once processing is initiated, an operation cannot be interrupted, and concurrency is not allowed. The value $C_{ij} = C_{ik} + p_{ij}$ is a completion time in operation u_{ij} in relation $u_{ik} \rightarrow u_{ij}$. p_{ij} is pre-set and the problem is only to find out the

completion time C_{ij} ($\forall$ u_{ij} $\in$O) which minimizes the objective function of JSSP as given below:

$$C_{\max} = \max_{all\ u_{ij} \in O}(C_{ij}) = \max_{u_{ik} \to u_{ij}}(C_{ik} + p_{ij}) \quad (7)$$

The JSP subjects to two constraints, known as the operation precedence constraint and machine processing constraint: The operation precedence constraint on the job is that the order of operations of job is fixed and the processing of an operation cannot be interrupted and concurrent, as given in Eqn.8.

$$C_{ij} \geq C_{kj} + p_{ij}, \quad \text{operation } u_{kj} \text{ is finished before } u_{ij} \quad (8)$$

The machine processing constraint is that only a single job can be processed at the same time on the same machine, as given in Eqn. 9.The operations must be assigned to the time intervals in such a way that once an operation is started it must be completed.

$$\neg \exists k(u_{ip} \to u_{ik} \vee u_{ik} \to u_{iq}), \text{ when } u_{ip} \to u_{iq},\ k \neq p \wedge k \neq q \quad (9)$$

Job Shop Scheduling benchmark problems from OR library are used in this paper to test the robustness of the proposed harmony BFO algorithm. In this research work, three algorithms, generic BFO framework, harmony search improvised BFO proposed by us termed as HBSA are implemented. In addition, the Ant Colony Optimization algorithm is also implemented to solve JSSP for the purpose of comparing the experimental results.

d) Implementation results

This section analyzes the result of the implementation of BFO, our proposed HBSA and ACO algorithms in solving JSSP benchmark instances . There are totally 82 test instances available in the OR library [7][28]. This study has chosen 22 problems of varying sizes from *abz and la* instances as listed in Table 1.

Table 1: JSSP instances used in this work

Instance	Size	Instance	Size
la 01	10 x 5	la 08	15 x 5
la 02	10 x 5	la 09	15 x 5
la 03	10 x 5	la 10	15 x 5
la 04	10 x 5	la 21	15 x10
la 05	10 x 5	la 24	15 x10
abz 5	10 x10	la 25	15 x10
abz 6	10 x10	la 27	20 x10
la 19	10 x10	la 29	20 x10
la 20	10 x10	abz 7	20 x15
la 06	15 x 5	abz 8	20 x 15
la 07	15 x 5	abz 9	20 x 15

The constant values initialized for the Harmony BSA are, ρ*=0.1,* β*=1.0,* α*=0.1, q0=0.8,* τ*=0.5,HMS=5, HMCR=0.9, PAR=0.4, NVAR=5, low=0, high=4 and BW=0.2.* Also the range of values chosen for harmony improvisation was *2.0-5.0, 3.0-6.0, 1.0-3.0, 2.0-3.0 and 1.0-4.0.*The result obtained by proposed Harmony BSA was compared with the results of BFO and ACO algorithms. It could be observed that our proposed method have achieved remarkably better optimization levels, almost equivalent to the best known optimal values obtained so far. The computational results are given below in table 2.

Table 2 : Optimality Comparison for benchmark JSSP instances

Instance	ACO	BFO	HBSA	Best-so-far
la 01	759	787	**666**	666
la 02	803	692	**668**	668
la 03	718	639	**624**	617
la 04	711	641	**614**	604
la 05	672	593	**593**	593
abz 5	1410	1323	**1246**	1234
abz 6	1046	1012	**956**	943
la 19	1033	926	**854**	842
la 20	1059	965	**959**	902
la 06	950	926	**926**	926
la 07	996	923	**903**	890
la 08	980	877	**873**	863
la 09	977	954	**951**	951
la 10	988	958	**958**	958
la 21	1324	1247	**1107**	1053
la 24	1241	1102	**942**	935
la 25	1242	1147	**978**	977
la 27	1610	1455	**1306**	1269
la 29	1481	1409	**1239**	1195
abz 7	837	787	**784**	668
abz 8	816	822	**792**	687
abz 9	921	856	**840**	707

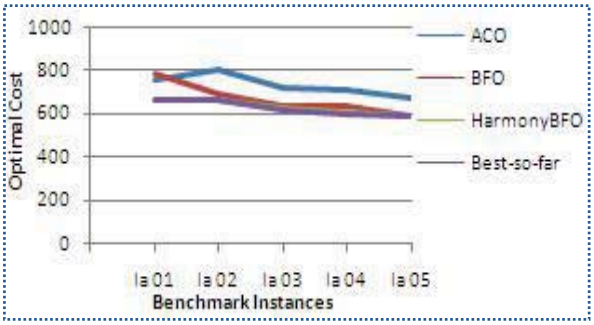

Fig. 2 : Optimal cost comparison of 10x5 size instances

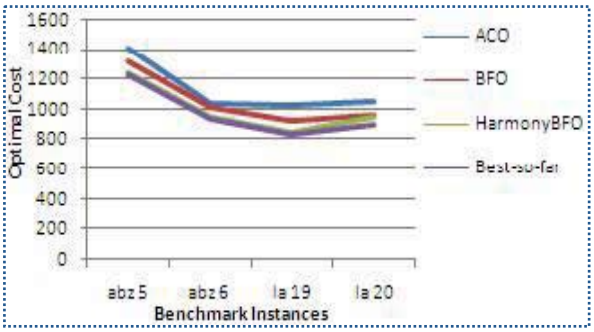

Fig. 3 : Optimal cost comparison of 10x10 size instances

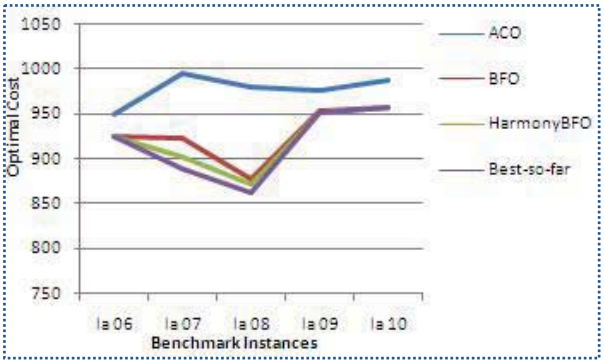

Fig. 4 : Optimal cost comparison of 15x5 size instances

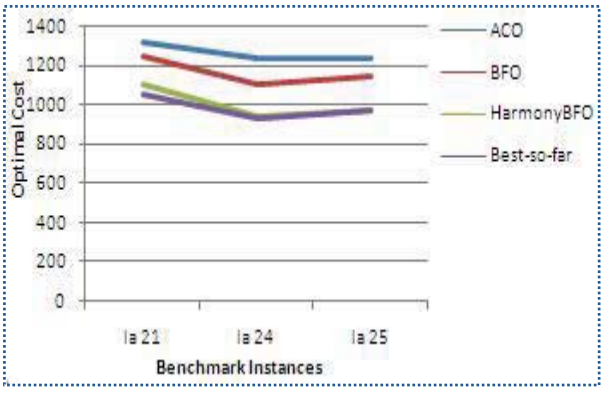

Fig. 5 : Optimal cost comparison of 15x10 size instances

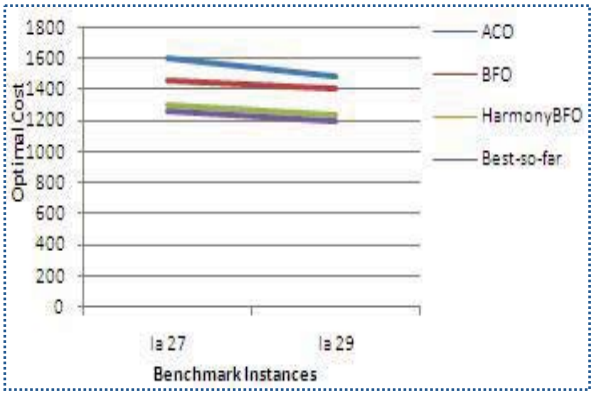

Fig. 6 : Optimal cost comparison of 20x10 size instances

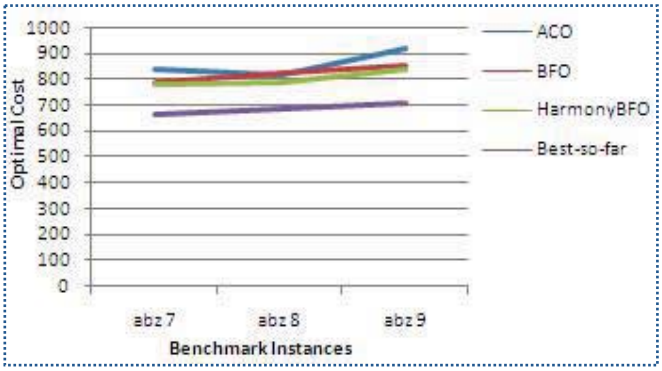

Fig. 7 : Optimal cost comparison of 20x15 size instances

From Fig.2, Fig.3, Fig.4, Fig.5, Fig.6 and Fig.7, it was clearly understandable that our proposed Harmony BSA algorithm gave the best optimal cost for all the chosen instances when compared with ACO and BFO algorithms. Also, it was observed that in most of the cases, our algorithm exactly matches the best known optimal values presented in the benchmark literature.

IV. Suggestions and Future Work

Scheduling Problems represent a rich domain in discrete optimization. Due to problem complexity many of these problems cannot be solved using traditional operations research techniques. In this paper, one of the most significant and complex classes of combinatorial optimization problems; the Job Shop Scheduling Problem was studied and implemented for analyzing the effectiveness of hybrid Harmony based Bacterial Swarming Algorithm. The JSSP determines a sequence for placing the jobs on the machines that optimizes a given evolution measure. The ability of the proposed Harmony BSA algorithm was investigated through the performance of several runs on 22 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 HBSA for JSSP are much better than ACO and BFO algorithms

and highly comparable to the best-so-far results obtained for the benchmark instances.

The proposed Hybrid Harmony BSA method can be further improved to implement the same for still larger benchmark instances that are found in the OR library. The problem of efficiently scheduling jobs on several machines is an important consideration when attempting to make effective use of a multi-machines system such as a Flexible Job Shop Scheduling system, which can also be developed using this proposed technique. This method also has high scope for modification for solving other type of assignment problems such as Quadratic Assignment Problems (QAP), Frequency Assignment Problems (FAP) and other types of scheduling problems, both static and dynamic cases. This work can be extended by implementing other local search techniques and testing the features to solve combinatorial optimization problems such as Vehicle routing, Scheduling, Travelling Salesman Problem, Bin packing and so on.

Acknowledgement

This research work is a part of the minor research project funded by University Grants Commission, India. The authors would like to thank the UGC for their financial support and encouragement in carrying out this research.

References Références Referencias

1. Al-Betar M.A and Khader A.T, *A harmony search algorithm for university course timetabling*, Springer Science+Business Media, LLC, 2010.
2. Batista B.M., José A., Perez M. and Vega J.M.M., *Nature-inspired Decentralized Cooperative Metaheuristic Strategies for Logistic Problems*, 2006.
3. Das T. K., and Venayagamoorthy G. K., *Bio-inspired algorithms for the design of multiple optimal power system stabilizers: SPPSO and BFA*. IEEE Transactions on Industry Applications, 2008.
4. Dasgupta S., Biswas A., Das S., Panigrahi B.K. and Abraham A., *A Micro-Bacterial Foraging Algorithm for High-Dimensional Optimization,* 2008.
5. David Applegate, William Cook., *A Computational Study of the Job-Shop Scheduling Problem*, 1991. (best so far ref)
6. Dorigo M., Blum C., *Ant colony optimization theory: A survey*, Theoretical Computer Science, 2005.
7. E. Taillard., *Benchmarks For Basic Scheduling Problems*, 1989.
8. Geem Z.W, *School Bus routing using Harmony Search*, GECCO 2005, 2005.
9. Hanif Khan F., Khan N., Inayatullah S., and Nizami S.T, *Solving TSP problem by using Genetic Algorithm*, International Journal of Basic & Applied Sciences IJBAS, 2009.
10. James Montgomery, Card Fayad and Sarja Petrovic., *Solution Representation for Job Shop Scheduling Problems in Ant Colony Optimization,* 2005.
11. Kim D.H, Abraham A. and Cho J.H, *A hybrid genetic algorithm and bacterial foraging approach for global optimization*, Information Sciences 177, 2007.
12. Lorpunmanee .S, Noor Sap .M, Abdullah .A.H, and Chompoo-Inwai .C, *An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment*, World Academy of Science, Engineering and Technology, 2007.
13. M. Tamer Ayvaz, *Application of Harmony Search algorithm to the solution of groundwater management models*, Advances in Water Resources 32, Elsevier, 2009.
14. Pan Q., Suganthan P.N., Liang J.J. and Tasgetiren M.F., *A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem*, Expert Systems with Applications 38, 2011.
15. Ponnambalam.S.G, Jawahar.N and Girish B. S, *An Ant Colony Optimization algorithm for Flexible Job shop scheduling problem*, 2008.
16. Reimann M., Shtovba S., and Nepomuceno E., *A hybrid ACO-GA approach to solve Vehicle Routing Problems*, 2005
17. Riganti Fulginei F., and SALVINI A., *Bacterial Chemotaxis Algorithm for Load Flow Optimization*, Proc. of the 5th WSEAS/IASME Int. Conf. on Electric Power Systems, 2005.
18. S. Subramanian and S. Padma., *Bacterial Foraging Algorithm Based Multiobjective Optimal Design of single phase Transformer*, 2011.
19. Shen H., Zhu Y., Zhou X., Guo H. and Chang C., *Bacterial Foraging Optimization Algorithm with Particle Swarm Optimization Strategy for Global Numerical Optimization*, 2008.
20. Tang W. J., Wu Q. H. and Saunders J. R., *Bacterial Foraging Algorithm For Dynamic Environments*, IEEE Congress on Evolutionary Computation, 2006.
21. Weise T., *Global Optimization Algorithms-Theory and Application*, v2nd Edition, 2003.
22. Wu1 C., Zhang N., Jiang J., Jinhui Yang J., and Liang Y., *Improved Bacterial Foraging Algorithms and Their Applications to Job Shop Scheduling Problems*, Springer-Verlag Berlin Heidelberg, 2007.
23. Yang X.-S., "*Harmony Search as a Metaheuristic Algorithm", in: Music-Inspired Harmony Search Algorithm: Theory and Applications*, Studies in Computational Intelligence, Springer Berlin, 2009.
24. Zhang J, Hu X, Tan X, Zhong J.H and Huang Q., *Implementation of an Ant Colony Optimization technique for job shop, scheduling problem*, Transactions of the Institute of Measurement and Control 28, 2006.

25. Zhou .R, Lee .P and Andrew Y.C. Nee, *Applying Ant Colony Optimization (ACO) algorithm to dynamic job shop scheduling problems*, Int. J. Manufacturing Research, 2008.
26. Zong Woo Geem, M. Fesanghary, Jeong-Yoon Choi, M. P. Saka, Justin C. Williams, M. Tamer Ayvaz and A. Vasebi, *Recent Advances in Harmony Search, Advances in Evolutionary Algorithms*, ISBN 978-953-7619-11-4, 2008.
27. Zou D., Gao L., Li .S, Wu J. and Wang X., *A novel global harmony search algorithm for task assignment problem*, The Journal of Systems and Software 83, 2010.
28. http://people.brunel.ac.uk/~mastjjb/jeb/info.html