

Performance comparison of some evolutionary algorithms on job shop scheduling problems

S.K.Mishra¹, and C S P Rao²

¹CEO, Bhramos, New Delhi, India

²Department of Mechanical Engineering, National Institute of Technology, Warangal, Telangana, India-506004

E-mail: mishrasudhir@gmail.com

Abstract: Job Shop Scheduling as a state space search problem belonging to NP-hard category due to its complexity and combinational explosion of states. Several naturally inspire evolutionary methods have been developed to solve Job Shop Scheduling Problems. In this paper the evolutionary methods namely Particles Swarm Optimization, Artificial Intelligence, Invasive Weed Optimization, Bacterial Foraging Optimization, Music Based Harmony Search Algorithms are applied and find tuned to model and solve Job Shop Scheduling Problems. To compare about 250 Bench Mark instances have been used to evaluate the performance of these algorithms. The capabilities of each these algorithms in solving Job Shop Scheduling Problems are outlined.

1. Introduction

Generally, Job Shop Scheduling Problem (JSSP) is an operational sequencing problem to process n jobs on m machines in a given sequence so as to optimally utilize the resources by complete processing of all jobs in a minimum possible time. JSSP has belongs to the category of NP hard problems where the search space of the problem is $(n!)m$. Several naturally inspired evolutionary techniques / methods have recently been developed to address these problems to get near optimal solutions in a reasonable time period thus several unsolved / difficulty to solve JSSPs became target for many researchers.

Some Traditional Optimization Techniques includes Priority Dispatch Rules, Efficient Methods solvable in polynomial times. Enumerative methods and Mathematical formulation like Linear Programming, Mixed Linear Programming, Lagrangian Relaxation, Branch and Bound and Disjunctive Graph techniques etc. the research has been accelerated with applications of Nontraditional Techniques. Application of Artificial Intelligence, insertion Algorithms, Bottleneck Heuristics, Neural Networks, Expert Systems and Local Search algorithms. Application of Evolutionary Algorithms for scheduling found from mid 80s to till date. Genetic Algorithm, Variation in GAs, PSO, ACO, BCO, Memetic Algorithm, Immune Algorithm and several Hybrid Algorithms. The BFO algorithms, HS algorithms, IWO have been found applied in JSSP in the recent years.

There is an increase in use of Non Traditional Method for JSSP compared to traditional method availability of computational time. For the same cost and time non traditional methods yield better solutions compared to traditional methods.



Hence, in this paper some classical evolutionary algorithms namely Particle Swarn Optimization, Article Immune System, Invasive weed optimization (IWO), Bacterial Foraging Optimization (BFO) and Music Based Harmony Search principles are applied and fine-tuned the mechanisms to model and solve JSSP. Several Bench Mark instances available in OR library were thoroughly tested to prove the efficiency of the proposed methods. Two different populations were considered for these algorithms. One random population and the other is selective population.

- i. Random Population (RP): The initial population is randomly generated and applied to the algorithm procedures and let us call such methods as PSO with RP, HSPO with RP, AIA with RP, HAIA with RP, MBHS with RP, IMBHS with RP, BFO and IWO with RP.
- ii. Selective Population: The initial Populations are generated using priority dispatching rules. A priority dispatching rule is a simple heuristic, based on some rules, specifies the priority of operations of jobs to be processed. 10 initial schedules i.e., populations are generated using 10 commonly used priority dispatching rules given in the Table 1 let us call these methods as PSO with SP, HSPO with SP, AIA with SP, HAIA with SP, MBHS with SP, IMBHS with SP, BFO with SP and IWO with SP.

EXPRESSION	DESCRIPTION
Shortest Processing Time (SPT)	The job with shortest processing time on machines are selected first $p_i \leq p_{i+1} \leq p_{i+2} \leq p_{i+3} \leq \dots \leq p_n$ $p_i < p_{i+1} < p_{i+2} < \dots < p_n$
Longest Processing Time(LPT)	The job with longest time on machines selected. $p_i \geq p_{i+1} \geq p_{i+2} \geq \dots \geq p_n$
Minimum Slack Time Per Operation(MINSOP)	Time remaining until the due date – Processing time Remaining
Minimum Due Date(MINDD)	The jobs with earliest due date are processed first $D_i \leq D_{i+1} \leq D_{i+2} \leq \dots \leq D_n$
Critical Ratio(CR)	Remaining due date/Remaining processing time
Most work remaining (MWKR)	Select the job of the most work remaining to be processed first.
Least work remaining(LWKR)	Select the job of the least work remaining to be processed first.
Shortest remaining Minimum Processing Time(SRMPT)	Min(processing time remaining- minimum processing time)
Longest remaining Maximum Processing Time(LRMPT)	Max(processing time remaining- maximum processing time)
RANDOM(random selection)	Select the job to be processed randomly.

Table 1. List of Priority Dispatching rules used for the initial population.

The coding of these algorithms is done in MATLAB, optimized by speed, and run on Intel Core2Duo T6400 @ 2.00GHz and each algorithm was made to run 30 times on each problem of 250 Bench Mark Problem Instances. In this chapter the results obtained by executing the above algorithm on 250 Bench Marking Problems are reported only.

2. Benchmark problems:

In the field of scheduling of Jobs in Machine shop, one needs to test on the bench mark problems. To compare the various techniques and algorithms these benchmark problems are available in the literature, Fisher and Thompson, 1963 (FT) have formulated 3 problems of 3 different sizes: 6×6, 10×10, 20×5; Lawrence, 1984 (LA) 40 problems of 8 different sizes: 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10 and 15×15; Adams Balas & Zawak, 1988 (ABZ) 5 problems of 2 different sizes: 10×10, 20×15; Applegate and Cook, 1991 (ORB) 10 problems of 10×10 size; Storer, Vaccari & Wu, 1992 (SWV) 20 problems of 3 different sizes: 20×10, 20×15, 50×10; Yamada and Nakano, 1992 (YN) 4 problems of 20×20 size; Taillard, 1993 (TA) had grouped 80 problems of 8 different sizes: 15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20; Demirkol, Mehta & Uzsoy, 1998 (DMU) formed 80 problems of 8 different sizes: 20×15, 20×20, 30×15, 30×20, 50×15, 50×20, 100×20; Jacques Carlier, (CAR) listed 8 problems of 8 different sizes: 11×5, 13×4, 12×5, 14×4, 10×6, 8×9, 7×7, 8×8; and are freely available. In this paper all the known 250 benchmark problems of JSSP have been taken into consideration for testing with the algorithms being developed.

3. Algorithms Developed:

3.1. Particle Swarm optimization (PSO) Algorithm:

P.S.O is one of the best evolutionary techniques for unconstrained continuous optimization. Being developed by Kennedy and Eberhart (1995). The social behaviors of birds (particles) for foraging is imitated in this algorithm. Particles move toward the *pbest* position and the particles (Swarm) in *pbest* tries to move to *gbest* position in each cycle. The *pbest* position is an opportunity to reach *gbest* position. The *gbest* position may be the best position in the swarm. Let k^{th} particle in dimension j has velocity v_{kj} and position x_{kj} and the velocity and position of k^{th} particle in dimensions particles can be updated by the following equations:

$$v_{kj} = wXv_{kj} + c_1Xrand_1X(pbest_{kj} - x_{kj}) + c_2Xrand_2X(gbest_j - x_{kj}).....(3.1)$$

$$x_{kj} = x_{kj} + v_{kj}.....(3.2)$$

In Equations (3.1) and (3.2), The other terms in the above are follows :

W = initial weight to exploration and exploration of search.

$C1, C2$ = random variable between 0-1.

The process of working of PSO is shown in Figure.1.

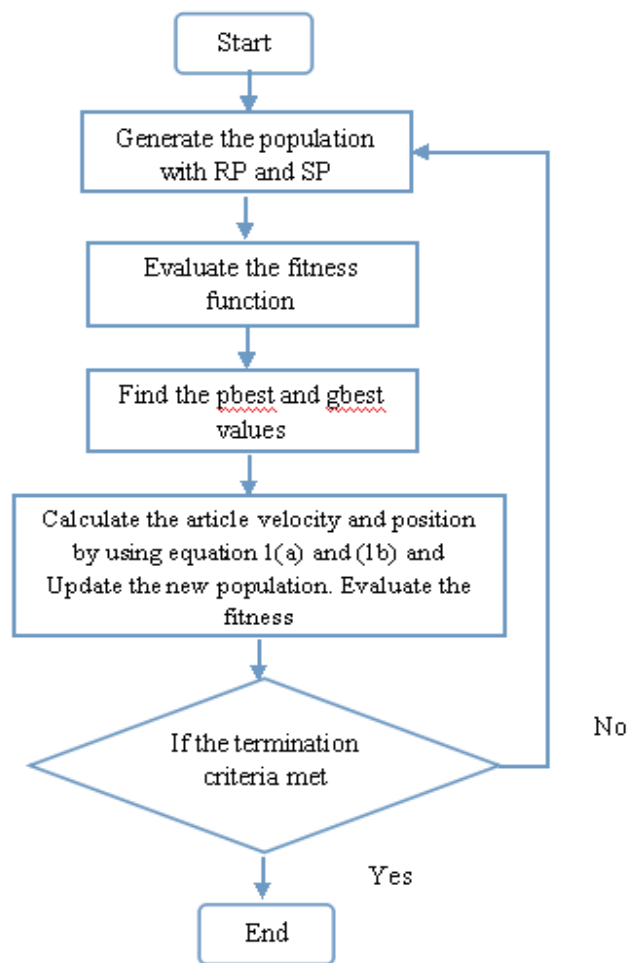


Figure.1 Flowchart for Particle Swarm Optimization

3.2. Hybrid Particle Swarm Optimization (HPSO) Algorithm:

PSO is, an evolutionary search algorithm; it may be trapped in local and scarcely reach global solution at the end of a run. PSO may fail to search the optimal solution in case of complex and complicated problem heuristic. The original PSO was designed for a continuous solution space. Simulated Annealing (SA) has certain probability and may avoid being trapped in a local optimum and different cooling schedules will control the search. By combining PSO and SA, we developed a general and fast converging hybrid algorithm called Hybrid Particle Search Optimization, such HPSO can be applied to many combinatorial optimization problems such as JSSPs. The psedudo code of HPSO is given on Figure.2.

Begin

Step 1: Initialization was done by both RP and SP

1) PSO

- ☐ Initialize swarm, including swarm size, each particle's position and velocity;
- ☐ Evaluate the each particle fitness;
- ☐ Initialize gbest position with particle with the lowest fitness in the swarm;
- ☐ Initialize pbest position with a copy of particle itself;
- ☐ Give initial value: W_{max} , W_{min} , $C1$, $C2$ and generation=0;

2) SA

```

Determine To,
Tend, B.
Step2: Computation
1) PSO
    While( the max of generation is not met)
    Do {
        Generation ++;
        Generate next swarm by equation (1a) and (1b);
    Evaluation Swarm {
        Find new gbest and pbest
        Update gbest of the swarm and pbest of each particle
    }
    }
2) EA
    for gbest particle S of Swarm
    {
        Tk=To;
        While (Tk>Tend)
        Do {
            Generate a neighbor solution S1 from S by pair exchange method;
            Compute fitness of S1;
            Evaluate S1 {
                 $\Delta = f(S1) - f(S)$ ;
                If ( $\min[1, \exp(-\Delta/Tk)] > \text{random}[0,1]$ ) {Accept S1}
                update the best solution found so far if possible;
            }
             $T_k = B * T_k$ ;
        }
    }
Step 3: Output optimization results
End

```

Figure.2 Pseudo code for Hybrid Particle Swarm Optimization (HPSO)

3.3. Artificial Immune Algorithm (AIA):

Artificial Immune Systems are highly adaptive and mimics human Immune system being explored, expected applied to problem solving. The field of AIS was initially developed in 1986 by J. D. Farmer et al. in their paper called 'The Immune System, Adaptation and Machine Learning' and was followed by another paper by G.W. Hoffman called 'A Neural Network Model Based on the Analogy with the Immune System'. Artificial Immune Model is, a basic model with understanding of the Functioning of the human immune system is essential. The human immune system is characterized by its robust and adaptive nature to counter the infection. This mechanism is explained as if an infection (antigen) attack and defense mechanism (Antibody) counter attack in human body. The artificial immune system may be described with two principles of the immune system.

- i. Clonal selection principle
- ii. Affinity maturation principle

i. Clonal selection principle

Each schedule is considered as an auto body and it has the affinity forward antigens. Affinity value of each schedule is defined by the affinity function. The affinity function is

$$\text{Affinity}(p) = 1/\text{makespan}$$

Higher the affinity lower the makespan and visa versa. Further the cloning of antibodies is done based on to their affinity values. Therefore, there will be more clones of antibodies that have lower makespan values than those with higher makespan values in the new generated clone population.

ii. Affinity maturation principle

The affinity maturation principle is a two phased mutation procedure used for the generated clones.

- a. Inverse mutation
- b. Pair wise interchange mutation

a. Inverse mutation

Let i and j be randomly selected two positions in a sequences. A neighbour of s is obtained by inverting the sequence of jobs between i and j positions. If the makespan value of the mutated sequence (after inverse mutation) is smaller than that of the original sequence (a generated clone from an antibody), then the mutated sequence is stored in the place of the original one. Otherwise, the sequence will be further mutated with pair wise random interchange mutation.

b. Pair wise interchange mutation

Let i and j be randomly selected two positions in a sequence s . A neighbour of s is obtained by interchanging the jobs in positions i and j . If the makespan of the mutated sequence (after pair wise interchange mutation) is less than that of the original sequence, then store the mutated one in the place of the original one. In the case where the algorithm could not find a better sequence after the two-mutation procedure, then it stores the original sequence (generated clone).

Figure.3 show the flowchart of AIS algorithm for solving the job shop scheduling problem. The possible schedules are represented by integer-valued sequences of length n (jobs). Each sequence of n -Jobs is a String representing as antibody of the AIS. The algorithm goes up to solution by the evolution of these antibodies.

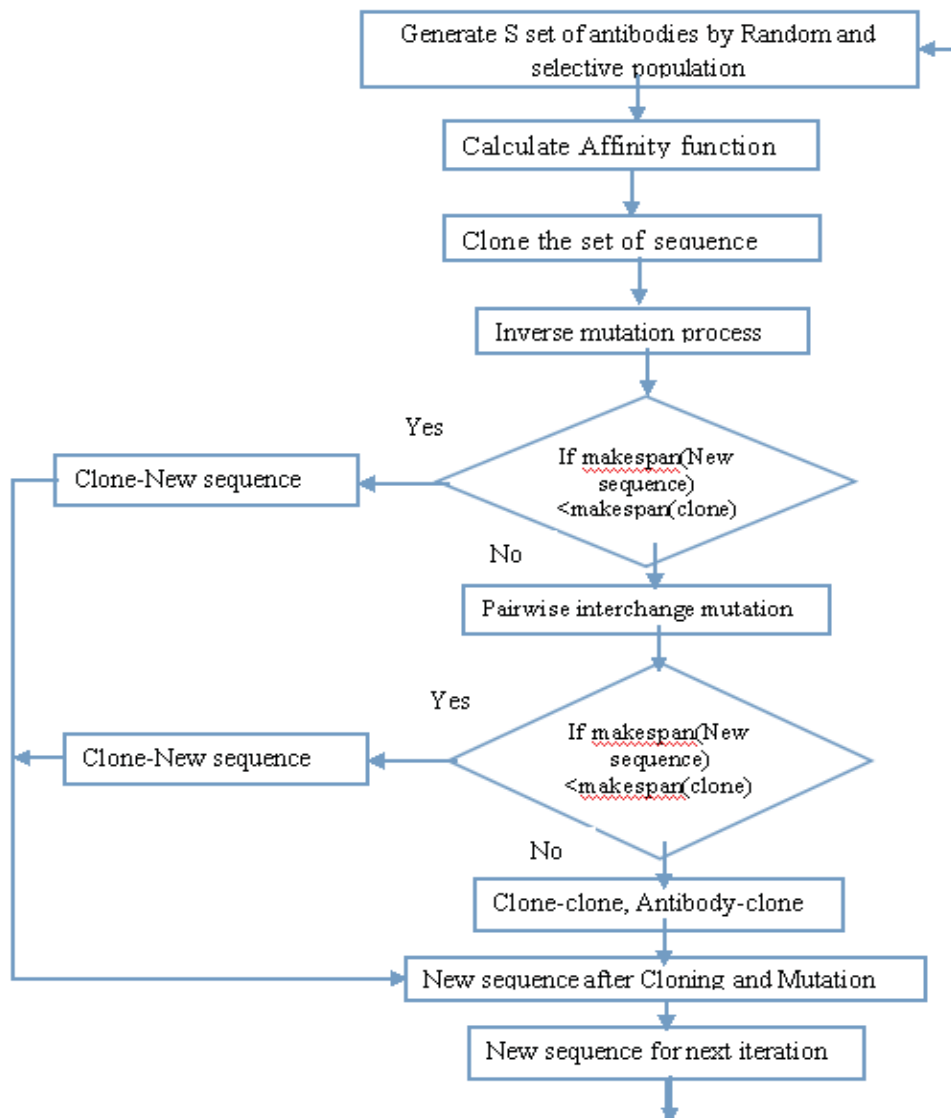


Figure.3 Flow chart of artificial immune system algorithm

3.4. Hybrid Artificial Immune Algorithm (HAIA):

The hybrid algorithm is a combination PSO and AIS algorithm for the JSSP. Hybrid Artificial Immune Algorithm adopts the antigens for finding optimum solutions efficiently. In this case antigen is a potential solution and the algorithm helps the antigens to evolve and generate better population thus giving rise to fitter antigens which represent competitive schedules.

To implement a basic artificial immune system, four decisions have to be made: encoding, similarity measure, selection and mutation. Once an encoding has been fixed and a suitable similarity measure is chosen, the algorithm will then perform selection and mutation, both based on the similarity measure, until stopping criteria are met. The pseudo code showing the main procedures of the algorithm is shown in Figure.4 To accelerate the convergence speed of the search algorithm, a neighborhood search mechanism using PSO concept is formulated especially JSSPs.

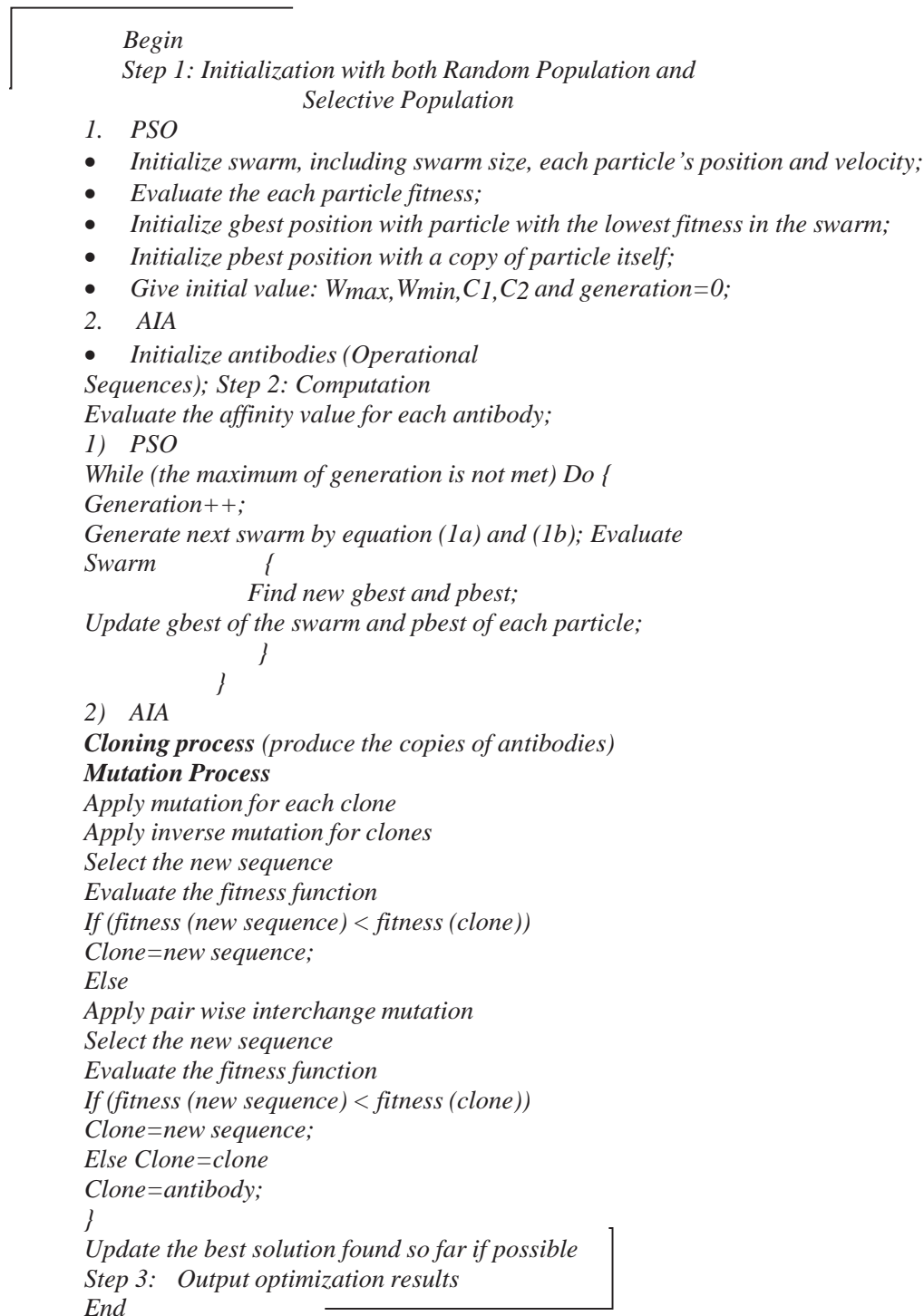


Figure.4 Pseudo code for Hybrid Artificial Immune Algorithm

3.5. Invasive Weed Optimization (IWO) Algorithm:

Invasive weed optimization (IWO) was first designed and developed by Mehrabian and Lucas. IWO is relatively a novel numerical stochastic optimization algorithm initiated by colonization of invasive weeds [11]. The algorithm is simple and powerful one in converging to optimal solution using basic properties, such as seeds, growth and competition, in a weed colony. A weed is a wild plant growing where it is not wanted; depending on the situation in any specified geographical area, any tree, vine, shrub or herb may qualify as a

weed. Weeds are strong and adaptive nature that contributes undesirable plants in agriculture. In D-dimensional search space, A weed is a potential solution for the proposed objective function. It is represented by $W = (w_1, w_2, \dots, w_m)$. Firstly, M weeds, called a population of plants, are initialized with random growth position, and then each weed produces seeds depending on its fitness and the colony's lowest fitness as well as highest fitness to simulate the natural survival of the fittest process. The number of seeds each plant produce increases linearly from minimum possible seed production to its maximum. The generated seeds are being distribution randomly in the search area by normal distribution with mean equal to zero and a variance parameter decreasing over the number of iterations. By setting the mean equal to zero, the seeds are distributed randomly such that they locate near to the parent plant and by decreasing the variance over time, the fitter plants are grouped together and inappropriate plants are eliminated over times.

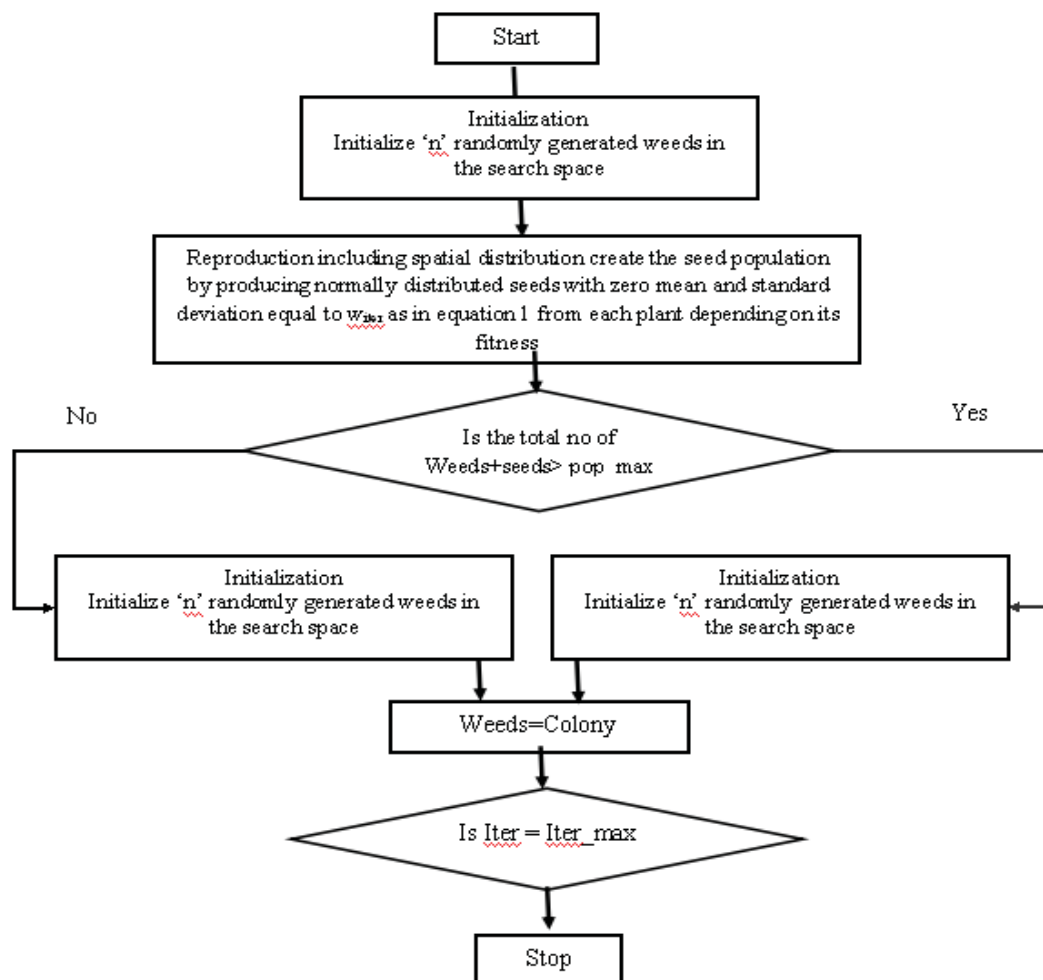


Figure.5 A Flowchart representation of IWO

The entire process is represented shown in Figures.5. Invasive Weed Optimization endorses the colonization process of weeds for finding optimal solutions efficiently. In this case weed is a potential solution as well as the algorithm helps the weeds to develop and generate better population thus giving rise to fitter weeds which represent competitive schedules.

3.6. Bacterial Foraging Optimization (BFO) Algorithm:

Bacterial Foraging Optimization (BFO) was first introduced by Passino inspired from the Swarm Intelligence. Bacteria search for nutrients in a manner to spread and maximize energy in a unit time. Individual bacterium do communicates with each others by sending signals. A bacterium takes decision based on searches and foraging

policy chemotaxis is the process, in which a bacterium moves by taking small steps for nutrients. BFOA is mimicking chemotactic movement of bacterium in the domain search space. The locomotion of real bacteria is achieved by a set of tensile flagella. Flagella help an E.coli bacterium and tumble or swim, which are two basic operations performed by a bacterium at the time of foraging. When they rotate the flagella in the clockwise direction, each flagellum pulls on the cell. That result in the moving of flagella independently and finally the bacterium tumbles with lesser number of tumbling whereas in a harmful place it tumbles frequently to find a nutrient gradient. Moving the flagella in the counter clockwise direction helps the bacterium to swim at a very fast rate. In the above-mentioned algorithm the bacteria undergoes chemotaxis, where they like to move towards a nutrient gradient and avoid noxious environment. Generally the bacteria move for a longer distance in a friendly environment.

A chemotactic step is performed by a tumble followed by a tumble or a tumble and a run. Let j , k , and l be the indexes for the chemotactic, reproduction and eliminate dispersal events respectively.

Also let $P(j, k, l) \{ (j, k, l) | i = 1, 2, \dots, S \}$ $i = q$ represent the position of each member in the population of the S bacteria at the j -th chemotactic step, k -th reproduction step, and l -th elimination-dispersal event. Here, let $J(i, j, k, l)$ denote the cost at the location of the i -th bacterium i . Note that we will interchangeably refer to J as being a “cost” and as being a nutrient surface. For original bacterium, S can be very large (e.g., $S = 109$), but $p = 3$. In our computer simulations, we will use much smaller population sizes and will keep the population size fixed. BFOA, however, allows $p > 3$ so that we can apply the method to a higher dimensional optimization problems. The process of BFO is shown in Figures.6.

3.7. Music Based Harmony Search (MBHS) Algorithm:

The harmony algorithm was first developed in 2001 by Geem, Kim and Loganathan is an efficient combinational optimization tool (Ingram and Zhang 2009).

Theory of Harmony Search has been successfully employed in engineering domains include Sudoku Puzzle (Geem, 2007), Tour Planning (Geem et al., 2005), Visual Tracking (Fourie et al., 2008), visual correspondence (Fourie et al., 2009), design of radar codes (Gil-Lopez et al., 2012), power and sub carrier allocation (Del Ser et al., 2011), design of wi-fi networks (Landa-Torres et al., 2012), single and bi objective localization (Manjarres et al., 2012; Manjarres et al., 2012a), structural design (Lee and Geem, 2004), water network design (Geem, 2006), vehicle routing (Geem et al., 2005), ground water modelling (Ayvaz, 2007), soil stability analysis (Cheng et al., 2008), satellite heat pipe design (Geem and Hwangbo, 2006), dam scheduling (Geem, 2007), ecological conservation (Geem, 2008), heat exchanger design (Fesanghary et al., 2009), face milling (Zarei et al., 2009), multi cast routing (Forsati et al., 2008) and several other fields.

The main five steps involved in MBH algorithm as described below:

The optimization of objective function is Max/Min $f(x)$ subject to $x_i \in X_i \quad i=1,2,3,4,\dots,N$.

where $f(x)$ = objective function,
 x = set of due some variables $[x_i]$.

Step 1: Initialize the problem and algorithm parameters.

Step 2: Initialize the harmony memory (HMS) i.e., no. of solution vectors in harmony memory.

Step 3: Improvise a new harmony or Harmony Memory Considering Rate (HMCR) where
 HMCR $[0, 1]$

Step 4: Update the harmony memory or Pitch Adjusting Rate (PAR) where PAR $[0, 1]$ **Step 5:** Check the stopping criterion [No. of improvisations].

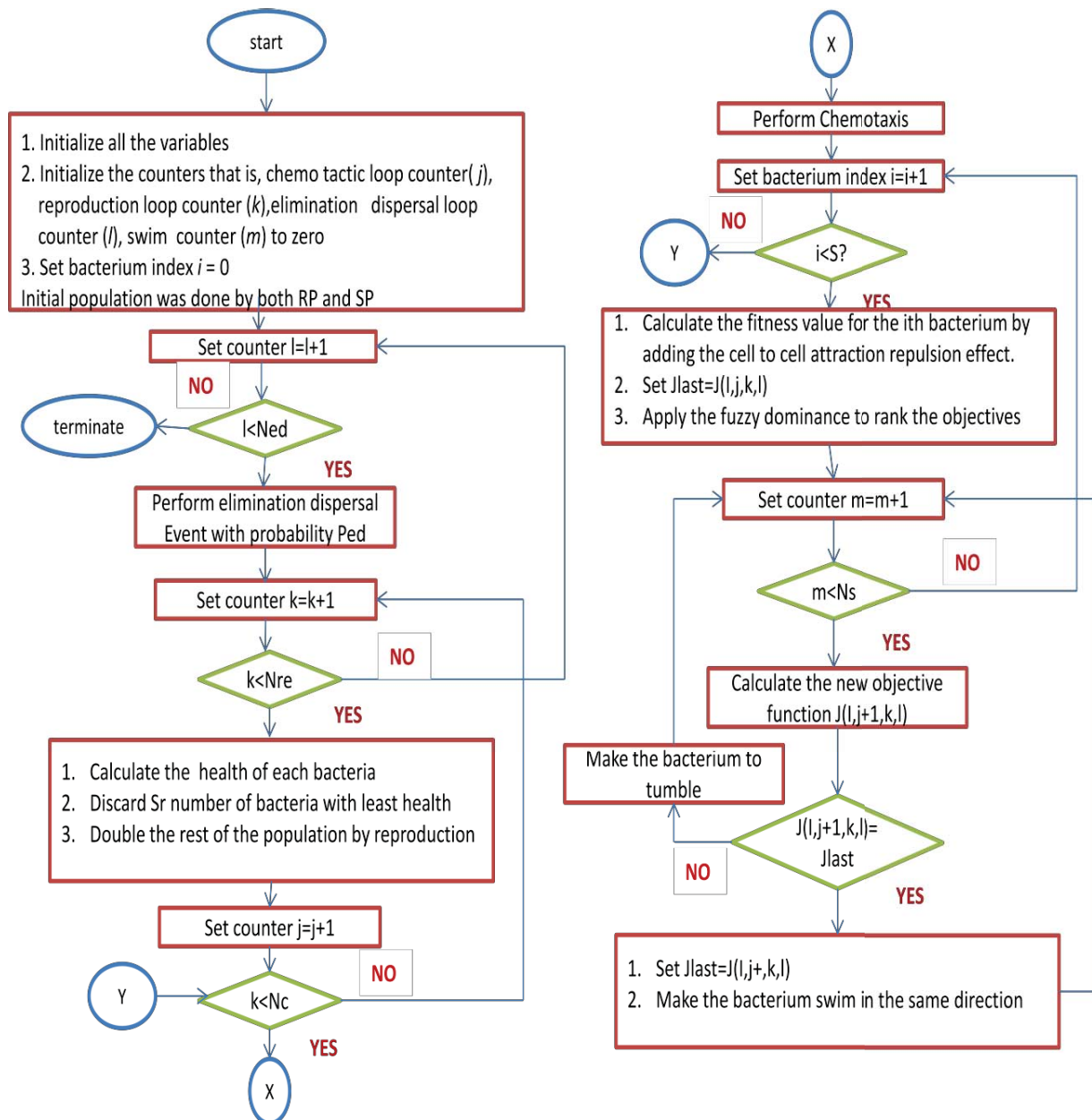


Figure.6 A Flow chart representation of BFO

Memory Consideration: When MBHS determines the value x_i' , it randomly picks any value x_{ij} from the HM with a probability of HMCr since $j = \{1, 2, \dots, HMS\}$.

$$x_i^1 \leftarrow \left\{ \begin{array}{l} x \in \{x_i^1, x_i^2, x_i^3, \dots, x_i^{HMS}\} \text{ with probability } HMCR \\ x_i^1 \in X_i \quad \text{with probability } (1-HMCR) \end{array} \right\}$$

Pitch Adjustment: Every component of the new harmony vector, $x' = (x_1', x_2', \dots, x_n')$, is examined to determine whether it should be pitch-adjusted. After the value x_i' is randomly picked from HM in the above memory consideration process, it can be further adjusted into neighboring values by adding certain amount to the value, with probability of PAR. This operation uses the PAR parameter, which is the rate of pitch adjustment given as follows:

$$x_i^1 \leftarrow \begin{cases} \text{yes with probability PAR} \\ \text{no with probability (1 - PAR)} \end{cases}$$

The value of (1-PAR) sets the rate of doing nothing. If the pitch adjustment decision for x_i^1 is yes, x_i^1 is replaced as follows:

$$x_i^1 \leftarrow x_i^1 \pm bw$$

Where, 'bw' is the arbitrary distance bandwidth for a continuous design variable.

In this step, pitch adjustment or random selection is applied to each variable of the New Harmony vector.

3.7.1. Improved Music Based Harmony Search (IMBHS) Algorithm:

The traditional MBHS algorithm uses fixed value for both PAR and bw. The PAR and bw values adjusted in the initialization step (Step 1) cannot be changed during new generations. The main drawback of this method is that this employs higher number of iterations to converge at an optimal solution.

The key difference between IMBHS and traditional MBHS method is in the way of adjusting PAR and bw. To improve the performance of the MBHS algorithm and eliminate the drawbacks associated with the fixed values of PAR and bw, the IMBHS algorithm uses variable PAR and bw in the improvisation step (Step 3). The PAR values change dynamically with

generation number expressed as follows:

$$PAR(gn) = PAR_{min} + \left[\frac{PAR_{max} - PAR_{min}}{NI} \right] \times gn$$

Where, PAR is the pitch adjusting rate for each generation, PAR_{min} is the minimum pitch adjusting rate, PAR_{max} is the maximum pitch adjusting rate, NI is the number of solution vector generations and gn is the generation number.

4. RESULTS AND ANALYSIS

4.1. Performance of Algorithms on FT (03) – Problems

It is found that while testing on FT problems (03), HPSO, IWO, MBHS, IMBHS, BFO, with both RP and SP are giving equal results with that of BKS. The solution with AIA and PSO with RP and SP are different on FT20 problem with BKS. PSO with SP had generated 1176 units of makespan against BKS of 1165 units. Similarly AIA with RP produced 1185 against 1165 of BKS. It is also found that the number of iterations for convergence is less with SP policy compared to RP policy. The results in the form of number of solutions equal to BKS on FT problems are shown in Figure.7.

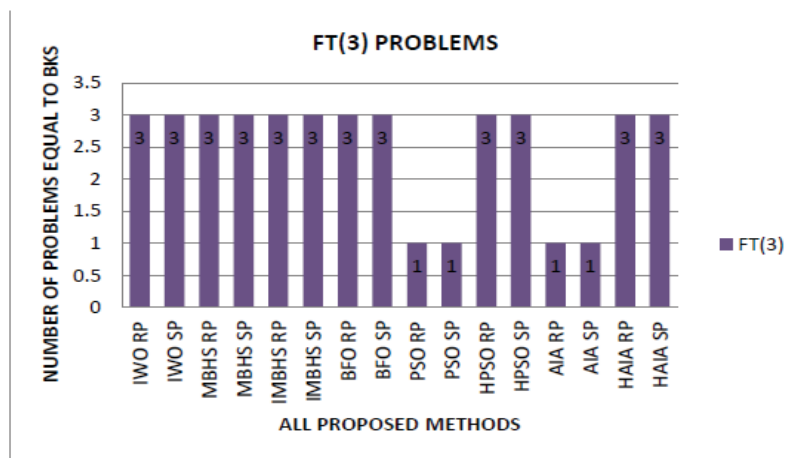


Figure.7 Comparison of FT, no. of problems equal to BKS by all proposed methods

4.2. Performance of Algorithms on LA (40) – Problems

It is found that HPSO, HAIA with SP and IWO, MBHS, IMBHS and BFO with both RP and SP are giving equal results of BKS. Out of 40 problems IWO, IMBHS, BFO with both RP and SP and AIA and HAIA with SP are giving equal results of BKS. Thus these algorithms are producing 100% results equal to BKS on all these LA series problems. PSO with both RP and SP are giving only 75% and 85% results equal to BKS respectively whereas HPSO with both RP and SP are giving 90% and 85% equal results with BKS respectively. AIA with RP and HAIA with RP are giving 82.5% matching with results with BKS. Figure.8 shows the result of number of problems equal to BKS by various algorithms.

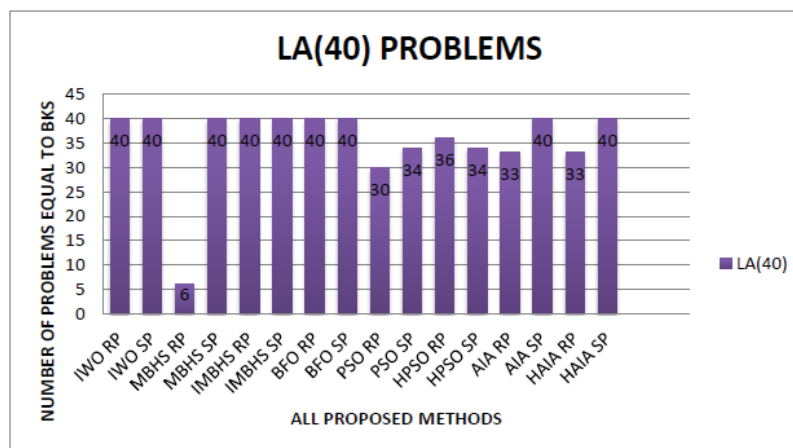


Figure.8 Comparison of LA, no. of problems equal to BKS by all proposed methods

4.3. Performance of Algorithms on ORB (10) - Problems

The makespan of each problems of ORB – Bench Mark Instance performed by each algorithm are presented.

From Figure 9, it is found that except AIA with RP remaining all proposed algorithms with both RP and SP are giving equal results with BKS i.e., 100% equal to BKS on ORB (10) problems. AIA with RP is giving only 30% results equal with BKS.

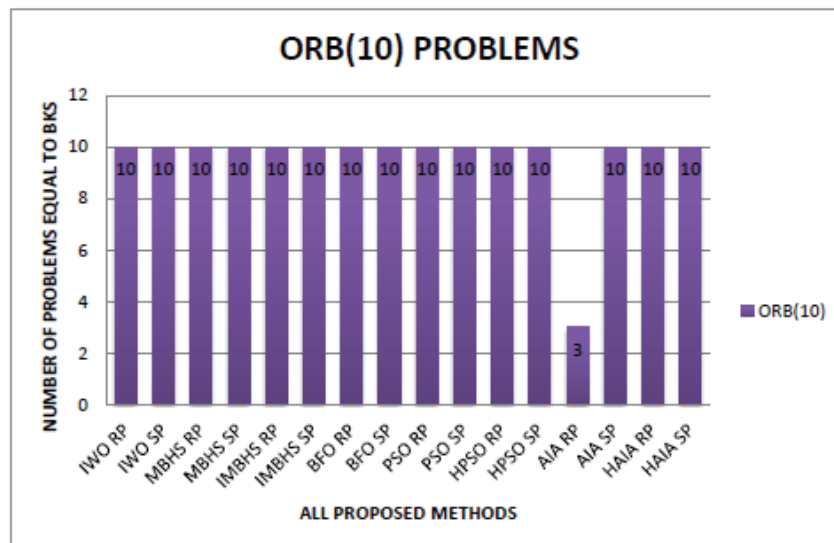


Figure.9 Comparison of ORB, no. of problems equal to BKS by all proposed methods

4.4. Performance of Algorithms on SWV (20) – Problems

It is found that while testing on SWV problems (20), all proposed methods with RP and SP are giving equal results with BKS (i.e., out of 20 problems tested equal results with BKS were produced on all 20 problems i.e., 100%). Figure.10 also shows this result.

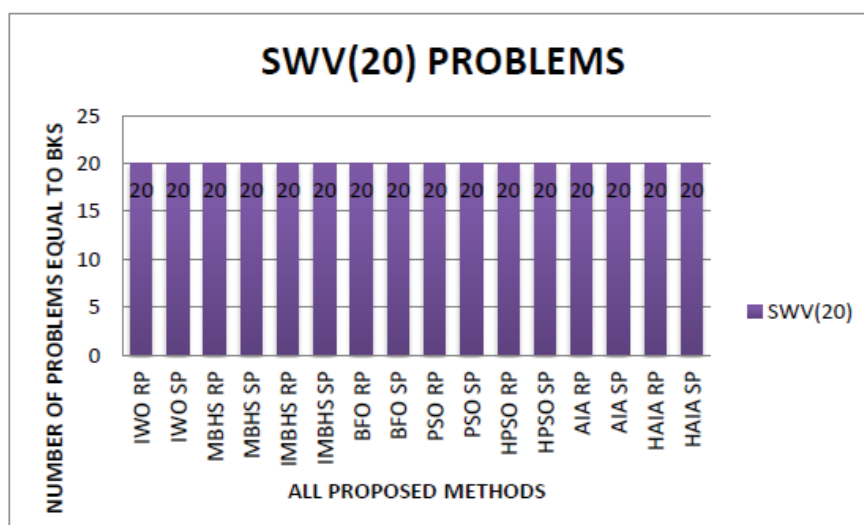


Figure.10 Comparison of SWV, no. of problems equal to BKS by all proposed methods

4.5. Performance of Algorithms on ABZ (05) – Problems

It is found that while testing on ABZ problems (10), IWO, MBHS, IMBHS, BFO, HPSO, HAIA with both RP and SP are giving equal results with BKS. PSO and AIA with SP are also giving equal results with BKS. PSO and AIA with RP only 30% results are equal to with BKS. The results were shown in Figure 11.

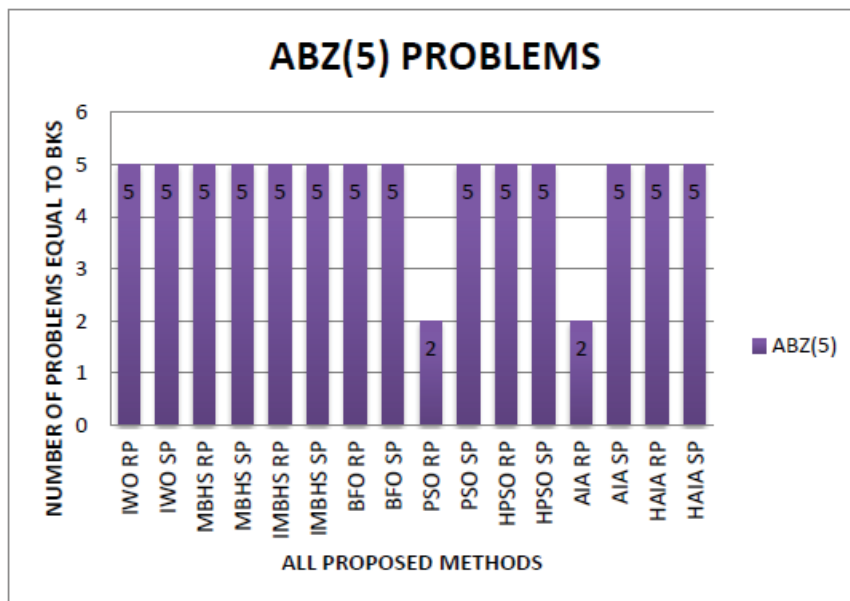


Figure.11 Comparison of ABZ, no. of problems equal to BKS by all proposed methods

4.6. Performance of Algorithms on YN (04) – Problems

It is found that while testing on YN problems (4), IWO, IMBHS, BFO, HPSO, HAIA with both RP and SP are giving equal results with BKS. MBHS, PSO, AIA with SP are also giving equal results with BKS. MBHS, PSO, AIA with RP are failed to give equal results to BKS. The results are also shown in Figure 12.

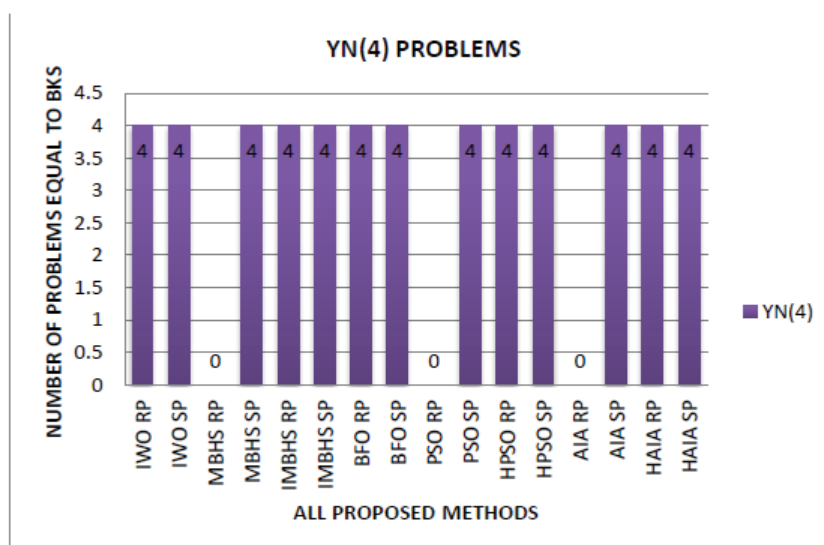
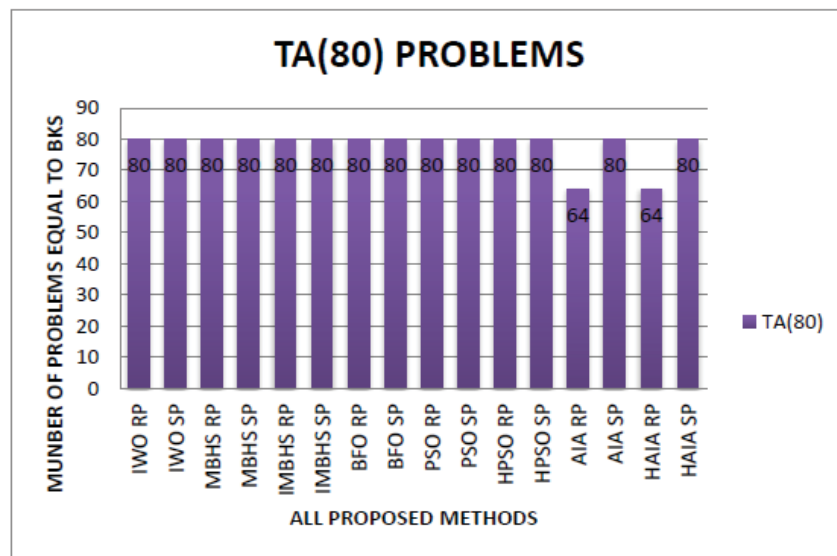
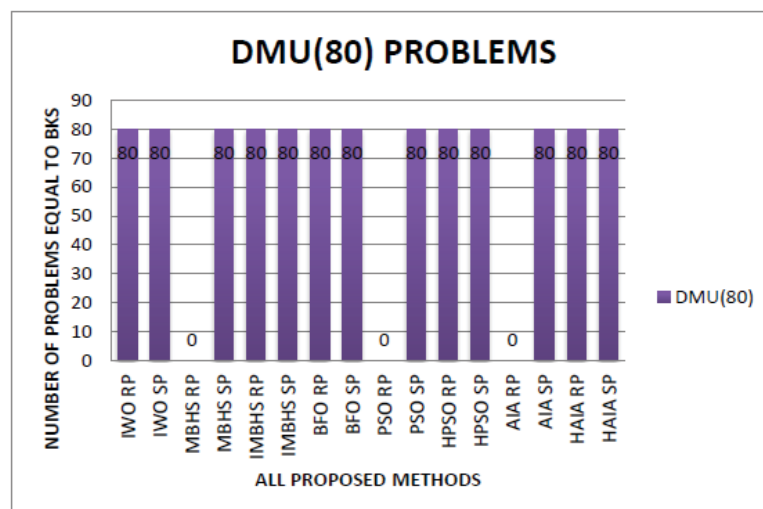


Figure.12 Comparison of YN, no. of problems equal to BKS by all proposed methods**4.7. Performance of Algorithms on TA (80) – Problems**

It is found that IWO, MBHS, IMBHS, BFO, PSO, HPSO with both RP and SP are giving 100% equal results with BKS. AIA and HAIA with SP is giving 100% equal results with that of BKS whereas in case of RP these are giving only 80% results equal with that of BKS. Figure 13 gives the number of problems equal to BKS.

**Figure.13** Comparison of TA, no. of problems equal to BKS by all proposed methods**4.8. Performance of Algorithms on DMU (80) - Problems**

It is found that IWO, IMBHS, BFO, HPSO, HAIA with both RP and SP are giving 100% equal results with BKS. MBHS, PSO, AIA with SP is also giving equal results with BKS, whereas in RP these methods have failed to give equal results with BKS. The same is shown in Figure 14.

**Figure.14** Comparison of DMU, no. of problems equal to BKS by all proposed methods

4.9. Performance of Algorithms on CAR (08) – Problems

It is found that IWO, IMBHS, BFO, HPSO, HAIA with both RP and SP are giving 100% equal results with BKS. MBHS, PSO, AIA with SP are also giving 100% equal results with SP whereas in RP these methods have failed to give equal results with BKS. The same is shown in Figure 15.

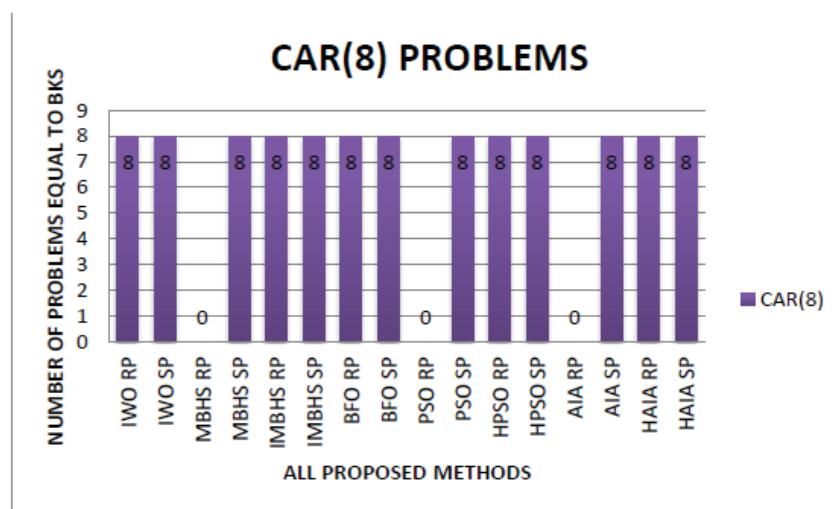


Figure.15 Comparison of CAR, no. of problems equal to BKS by all proposed methods

The consolidated results were shown in Table.2.

TOTAL NUMBER OF BENCH MARK INSTANCES = 250																								
Problems		FT (03)		LA(40)		ORB(10)		SWV(20)		ABZ(5)		YN(4)		TA(80)		DMU(80)		CAR(8)						
=	≠	% of =	=	≠	% of =	=	≠	% of =	=	≠	% of =	=	≠	% of =	=	≠	% of =	=	≠	% of =				
IMBHS RP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%			
IWO RP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%			
IWO SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%			
IMBHS RP	3	0	100%	6	34	15.0%	10	0	100%	20	0	100%	5	0	100%	0	4	0%	80	0	80	0%		
IMBHS SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
IMBHS RP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
IMBHS SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
IMBHS RP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
IMBHS SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
BFO RP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
BFO SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
PSO RP	1	2	33.3%	30	10	75%	10	0	100%	20	0	100%	2	3	40%	0	4	0%	80	0	80	0	0%	
PSO SP	1	2	33.3%	34	6	85%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
HPSO RP	3	0	100%	36	4	90%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
HPSO SP	3	0	100%	34	6	85%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
AIA RP	1	2	33.3%	33	7	82.5%	3	7	30%	20	0	100%	2	3	40%	0	4	0%	64	16	80%	0	80	0%
AIA SP	1	2	33.3%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	100%	
HAIA RP	3	0	100%	33	7	82.50%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	64	16	80%	80	0	100%
HAIA SP	3	0	100%	40	0	100%	10	0	100%	20	0	100%	5	0	100%	4	0	100%	80	0	100%	80	0	100%

Table 2. Comparison of % of Improvement table between RP and SP for different problems

From this Table 5.19, “=” represents the no. of problems equal to BKS, “#” represents the no. of problems not equal to BKS, “% of =” represents the % of Equivalence with the BKS.

Conclusions:

1. All the algorithms such as PSO, AIA, IWO, BFO, MBHS and hybridized methods such as HPSO, HAIA and IMBHS with RP and SP were tested on 250 Bench mark instances and found that Algorithms with selective initial population found better compared to random initial population.
2. The author has also tested IWO, IMBHS, MBHS and BFO on all existing Bench Mark Instances and found that IWO, IMBHS, MBHS and BFO were more stable and quickly converging though it is executed up to 1000, 5000 and 10000 iterations and yields best solutions equal to BKS. Hence IWO, IMBHS, MBHS and BFO may be considered excellent tools for addressing all the types of JSSPs and also resolved that they may be unique tools to solve all kinds of JSSPs.
3. The author had proposed hybrid methods using simulated annealing in PSO, PSO in AIA and improvement in MBHS algorithm and the developed hybrids are HPSO, HAIA, IMBHS algorithms and he found that the results were improved due to the proposed hybridization. The percentage of improvement is ranging from 45 - 50% with Algorithms with random population. Whereas hybrid effect is less observed with Algorithms with selected population. Hybridization improves the quality of solutions in general. This may be due to insertion of local search procedures within the global search being done by Algorithms.
4. From this study, it is claimed that HAIA, IWO, MBHS, IMBHS and BFO have emerged as unique and best tools to solve single objective JSSPs.

References:

- [1] Bagheri, M. Zandieh, I. Mahdavi, M. Yazdani (2010) An artificial immune algorithm for the flexible job - shop scheduling problem Future Generation Computer Systems. *Journal of Future Generation Computer Systems*. DOI: 10.1016/j.future.2009.10.004
- [2] A. Colorni, M. Dorigo et V. Maniezzo (1991) Distributed Optimization by Ant Colonies. *European conference on artificial life. Paris, France, Elsevier publishing*, 134–142.
- [3] El-Bouri, N. Azizi, S. Zolfaghari (2007) A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling. *European Journal of Operational Research*, Volume 177, Issue 3, Pages 1894-1910.
- [4] R.Mallahzadeh, H.Oraizi and Z.Davoodi-Rad (2008) Application of Invasive Weed Optimization technique for antenna configuration. *Progress in Electro-magnetic Research. PIER* 79, Pages 137-150
- [5] A.S Jain, S. Meeran Mascis, Dario Pacciarelli (1999) Deterministic Job-shop scheduling : Past , Present and Future. *European Journal of Operational Research*. Volume 113, Pages 390-434
- [6] Adil Baykasoğlu, Lale Özbakır, Türkay Dereli (2002) Multiple Dispatching Rule based heuristic for Multi- objective Scheduling of Job Shops using Tabu Search. *5th International Conference on Managing Innovations in Manufacturing*. Milwaukee, WI, USA
- [7] Aniruddha Basak, Siddharth Pal Swagatam Das, Ajith Abraham, Vaclav Snasel (2010) A modified Invasive Weed Optimization algorithm for time-modulated linear antenna array synthesis. *Evolutionary Computation (CEC), 2010 IEEE Congress on*. DOI: 10.1109/CEC.2010.5586276
- [8] D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, S. Salcedo-Sanz, R. Lopez-Valcarce (2012) On the Design of a Novel Two-Objective Harmony Search Approach for Distance and Connectivity-based Node Localization in Wireless Sensor Networks. *Engineering Applications of Artificial Intelligence*. Volume 26, Issue 2, Pages 669–676.
- [9] Blazewickz J (1996) The job shop scheduling problem: Conventional and new solutions techniques. *Eur J Oper Res* pp 931–30. DOI:10.1007/978-3-642-78910-6_43
- [10] Bou Shaala, Shouman, and Esheem.S (2012) Some Heuristic Rules for Job Shop Scheduling Problem *Industrial and manufacturing systems engineering*. Garyounis University, Libya.
- [11] Carlier J, Pison E (1989) An algorithm for solving the job shop problem. *Manage Sci* 35:164–176. <http://dx.doi.org/10.1287/mnsc.35.2.164>
- [12] N L de Castro, and Timmis J (2003) Artificial Immune Systems as a Novel Soft Computing Paradigm. *Soft Computing Journal*. vol. 7(7). DOI: 10.1007/s00500-002-0237-z
- [13] D.Y. Sha, Cheng-Yu Hsu (2006) A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering*. Volume 51, Issue 4, Pages 791-808.

- [14] Geem, Z. W. (2007) Optimal Scheduling of Multiple Dam System Using Harmony Search Algorithm. *Lecture Notes in Computer Science*.
- [15] Dasgupta, D (1999) An Overview of Artificial Immune Systems and Their Applications. *In Artificial Immune Systems and Their Applications*, Springer-Verlag, pp. 3 – 21.
- [16] Ferdinando Pezzella, Emanuela Merelli (2000) A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*. Volume 120, Issue 2. Pages 297-310.
- [17] M Fesanghary, E Damangir, I Soleimani (2009) Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search. *Applied Thermal Engineering*. Volume 29, Issues 5–6, Pages 1026–1031
- [18] Helena Ramalhinho Lourenço (1995) Job-shop scheduling: Computational study of local search and large-step optimization methods. *European Journal of Operational Research*. Volume 83, Issue 2, Pages 347-364.
- [19] K. Lee and Z. Geem (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering* 194, pp.3902-3933.
- [20] K. M. Passino (2002) Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67.
- [21] K. Steinhöfel, A. Albrecht, C. K. Wong (1999) Two simulated annealing-based heuristics for the job shop scheduling problem. *European Journal of Operational Research*, Volume 118, Issue 3(1), Pages 524-548.
- [22] M. Chandrasekaran . P. Asokan . S. Kumanan . T. Balamurugan . S. Nickolas (2006) Solving job shop scheduling problems using artificial immune system. *Springer-Verlag London Limited*. DOI: 10.1007/s00170-005-0226-3
- [23] M. Dorigo et L.M. Gambardella (1997) Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, volume 1(1), pages 53-66, 1997.
- [24] M.Mahdavi, M.Fesanghary, E.Damangir (2007) An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*. 188, pp.1567–1579.
- [25] M.Ramezani Ghalenoei, H.Hajmirsadeghi, C.Lucas (2009) Discrete Invasive Weed Optimization and its application to Co-operative multi-task assignment of UAVs. *Comin prac. 48th IEEE conference on Decision and Control*. DOI: 10.1109/CDC.2009.5400938
- [26] Ritwik Giri ,Aritra Chowdhury, Arnob Ghosh (2010) A Modified Invasive Weed Optimization for training of feed forward neural networks. DOI: 10.1109/ICSMC.2010.5642265
- [27] Runwei Cheng, Mitsuo Gen, Yasuhiro Tsujimura (1999) A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. *Computers & Industrial Engineering*, Volume 36, Issue 2. Pages 343-364.

- [28] S. D. Müller, J. Marchetto, S. Airaghi, and P. Koumoutsakos (2002) Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 16–29, 2002.
- [29] Geem, Z. W. and Hwangbo H. (2006) Application of Harmony Search to Multi- Objective Optimization for Satellite Heat Pipe Design. *Proceedings of US-Korea Conference on Science, Technology, & Entrepreneurship*. NJ, USA, Aug. 10-13 2006.
- [30] Siddharth Pal, Anniruddha Basak and Swagatam Das (2010) A Invasive Weed Optimization method based Multi-user detection for MC-CDMA Interference suppression over multiple-path fading channel. *IEEE* 978-422-6588-0/10/&25.00.
- [31] Xueni Qiu · Henry Y. K. Lau (2012) An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing*. Volume 25, Issue 3, pp 489–503. DOI: 10.1007/s10845-012-0701-2