IMPROVED MUSIC BASED HARMONY SEARCH (IMBHS) FOR SOLVING JOB SHOP SCHEDULING PROBLEMS (JSSPS)

Hymavathi Madivada[1] and C.S.P. Rao[2]

[1,2]Department of Mechanical Engineering, National Institute of Technology Warangal, Warangal – 506004.

hyma.madivada07@gmail.com csp_rao@rediffmail.com

ABSTRACT

In this paper, a new meta-heuristic for Job Shop Scheduling Problems (JSSPs) is presented. A Music Based Harmony Search algorithm (MBHS) is a recently developed algorithm which is conceptualized using the musical process of searching for a perfect state of harmony. It uses a stochastic search instead of a gradient search. Music Based Harmony Search algorithm (MBHS) and an Improved Music Based Harmony Search (IMBHS) algorithm were proposed to minimize the Makespan. The results are compared with Bench Mark Solutions (BKS) and it is found that both the methods performed better in terms of the quality of solution but in few problems IMBHS is performing better when compared to the MBHS method and BKS solutions. The results obtained in this study have shown that the proposed IMBHS algorithm can be used as a new alternative solution technique for finding good solutions to the JSSPs.

KEYWORDS

Metaheuristics, Job Shop Scheduling, Music Based Harmony Search, Improved Music Based Harmony Search

1. INTRODUCTION

A schedule is a time based allocation of the tasks to time intervals on the resources in job shop Industry. The optimal schedule is one sequence of tasks that minimizes the overall completion time. This is called the makespan. In the job shop scheduling problem is concerned with n jobs have to be processed on m different machines. Each job consists of a sequence of tasks that have to be processed during an uninterrupted time period of a fixed length on a given machine. So the maximum of completion times needed for processing all jobs, subject to the constraints that each job has a specified processing order through the machines and that each machine can process at most one job at a time. Job Shop Scheduling Problems (JSSP) belongs to class of NP-hard problems. In many Job Shop cases, the combination of goals and resources exponentially increases the search space, and thus the generation of good schedule is difficult because of a very large combinatorial search space. Several Job Shop Scheduling Problems in various industrial environments are combinatorial in nature and conventional mathematical tools have failed to produce optimum schedules. Thus JSSP is ever lasting field for good research.

The task of production scheduling consists in the temporal planning of the processing of a given set of orders. The processing of an order corresponds to the production of a particular product. It is accomplished by the execution of a set of operations in a predefined sequence on certain resources, subject to several constraints. The result of scheduling is a schedule showing the temporal assignment of operations of orders to the resources to be used. Each operation can be performed by some machines with different processing times. The difficulty is to find a good assignment of an operation to a machine in order to obtain a schedule which minimizes the total processing time of all jobs i.e., makespan.

Bruker [1] and Garey [2] show that the job shop scheduling is an NP-hard combinatorial problem. Because of the NP-hard characteristics of job shop scheduling, it is usually very hard to find its optimal solution, and an optimal solution in the mathematical sense is not always necessary in practice [3]. Researchers turned to search its near- optimal solutions with all kinds of heuristic algorithms [4]. Fortunately, the searched near optimal solutions usually meet requirements of practical problems very well.

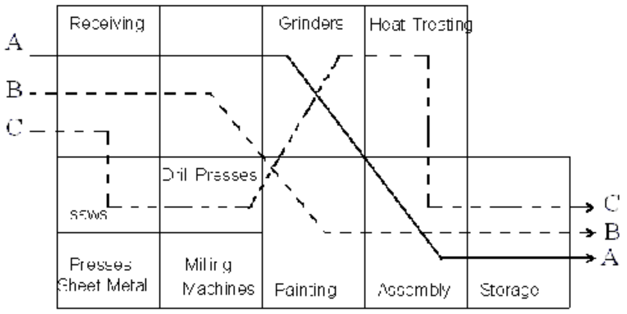

Figure 1: Schematic Diagram of Job Shop Problem

Carlier and Pison [5], Bruker [6] managed to generate optimal schedules using an algorithm and branch and bound approach, but only for small size problems. There- fore the effort has been directed by Blazewicz [7] to the development of efficient local search based heuristics to solve practical size problems.

Tabu search with bottleneck procedure (TSSB) was used to test the job shop scheduling benchmark problem instances [8] in the past work. The outcome of that TSSB procedure was compared with literature results [9]. It gave an optimum bound value for a moderate number of problems and with considerable amount of relative error.

1.1 Job Shop Scheduling Problem

Normally, the entire job shop scheduling problem consists of two types of constraints: sequence constraint and resource constraint [10]. The first type states that two operations of a job cannot be processed at the same time. The second type states that no more than one job can be handled on a machine at the same time. Job shop scheduling can be viewed as an optimization problem, bounded by both sequence and resource constraints. For a job shop scheduling problem, each job may consist of a different number of operations, subjected to some precedence restrictions. Commonly the processing orders of each job by all machines and the processing time of each operation are known and fixed. Once started operations cannot be interrupted.

Assume job i (i=1,2,...n) requires processing by machine k (k=1,2,... m) exactly once in its operation sequence (thus, each job has m operations). Let P_{ik} be the processing time of job i on machine k, X_{ik} be the starting time of job i on machine k, Q_{ijk} be the indicator which takes on a value of 1 if operation j of job i requires machine k, and zero otherwise. Y_{ihk} is the variable which takes on a value of 1 if job i precedes job h on machine k, and zero otherwise. The objective function is minimizing the Make-span, make-span being the maximum completion time of all jobs or the time taken to complete the last job on the last machine in the schedule, for the given job shop scheduling is:

Minimize $C = \sum_{k=1}^{m} Q_{imk} (X_{ik} + P_{ik}) \qquad (i = 1,2, \dots \dots \dots n) - - - \rightarrow (1)$
Subjected to the following constraints

a) The precedence of operations given by each job is to be respected.
b) Sequence Constraint: Each machine can perform at most one operation at a time
$\sum_{k=1}^{m} Q_{ijk}(X_{ik} \mid P_{ik}) \leq \sum_{k=1}^{m} Q_{i,j+1,k} X_{ik}, \quad (i = 1, \dots \dots n; j = 1, \dots \dots m \quad 1) \qquad , (2)$
ie., for a given job I, the (j+1)th operation may not start before the jth operation is completed.
c) Resource Constraint : The operations cannot be interrupted
$\left.\begin{array}{l} X_{hk} - X_{ik} \geq P_{ik} - (H + P_{ik})(1 - Y_{ihk}), \\ X_{ik} - X_{hk} \geq P_{hk} - (H + P_{hk})(Y_{ihk}), \end{array}\right\} where\, (i = 1, \dots \dots n; h = 1, \dots \dots n; k = 1, \dots \dots m) \quad - - - \rightarrow (3)$
Where H is a very large positive integer, chosen so that only one of the above constraints binding either for $Y_{ihk} = 1$ or for $Y_{ihk} = 0$.
d) The starting times of jobs are to be respected.

In this paper, the authors have employed Music Based Harmony Search (MBHS) algorithm developed by Geem et al in 2001 to model and solve JSSPs. The authors have also successfully implemented Improved Music Based Harmony Search (IMBHS) algorithm for JSSP. The two algorithms of MBHS and IMBHS first time applied and tested on many Job Shop Scheduling Problems Bench Mark Instances.

2. MUSIC BASED HARMONY SEARCH ALGORITHM (MBHSA)

The harmony search algorithm (Geem et al. 2001) is one of the most recently developed optimization algorithm and at a same time, it is one the most efficient algorithm in the field of combinatorial optimization (Geem 2009c). Since the emergence of this algorithm in 2001 by Geem et al., it attracted many researchers from various fields especially those working on solving optimization problems (Ingram and Zhang 2009). Consequently, this algorithm guided researchers to improve on its performance to be in line with the requirements of the applications being developed. HS imitates the natural phenomenon of musician's behavior when they cooperate the pitches of their instruments together to achieve a fantastic harmony as measured by aesthetic standards. This musicians prolonged and intense process led them to the perfect state. It is a very successful metaheuristic algorithm that can explore the search space of a given data in parallel optimization environment, where each solution (harmony) vector is generated by intelligently exploring and exploiting a search space (Geem 2009a). it has many features that make it as a preferable technique not only as standalone algorithm but also to be combined with other metaheuristics algorithms. Harmony Search as mentioned mimic the improvisation process of musician's with an intelligent way. The analogy between improvisation and optimization is likely as follows Geem (2010):

i. Each musician corresponds to each decision variable;

ii. Musical instrument's pitch range corresponds to the decision variable's value range;

iii. Musical harmony at a certain time corresponds to the solution vector at a certain iteration;

iv. Audience's aesthetics corresponds to the objective function.

Just like musical harmony is improved time after time, solution vector is improved iteration by iteration. Harmony search (HS) is a phenomenon – mimicking algorithm belongs to evolutionary class of metaheuristic algorithm, inspired by the harmony improvisation process of musicians. In HS metaheuristic, each musician is like decision variables plays (generates) a note (value) for finding a best harmony (optimization function). The HS algorithm has the following merits.

i. HS does not require differential gradients, thus it can consider discontinuous functions as well as continuous function.

ii. HS can handle discrete variables as well as continuous variables

iii. HS does not require initial value setting for the variables.

iv. HS is free from divergence

v. HS may escape local optima

vi. HS may overcome the drawback of GA’s building block theory which works well only if the relationship among variables in a chromosome is carefully considered. If neighbor variables in a chromosome have weaker relationship than remote variables, building block theory may not work well because of crossover operation. However, HS explicitly considers the relationship using ensemble operation.

vii. HS is a novel stochastic derivative applied to discrete variables, which uses musician’s experiences as a searching direction.

viii. Certain HS variants do not require algorithm parameters such as HMCR and PAR, thus novice users can easily use the algorithm.

Theory of Harmony Search has been successfully employed in engineering domains include Sudoku Puzzle [11], Tour Planning [12], Visual Tracking [13], visual correspondence[14], design of radar codes [15], power and sub carrier allocation [16], design of wi-fi networks [17], single and bi objective localization [18,19], structural design [20], water network design [21], vehicle routing [22], ground water modeling [23], soil stability analysis [24], satellite heat pipe design [25], dam scheduling [26], ecological conservation [27], heat exchanger design [28], face milling [29], multi cast routing [30] and several other fields.

In general Harmonic Search is executable in five steps as described below:

The objective function of optimization is Max/Min f(x) subject to $x_i \in X_i$ i=1,2,…,N. where f(x) is objective function, x is set of due some variables [x_i].

Step1: Initialize the problem and algorithm parameters

Consider an optimization problem which is described as:
Minimize the makespan i.e. the set of completion times for each operation {$C_{i,j,}$,k} 1≤i≤Xj, 1≤j≤N, 1≤k≤M which minimizes $C_{max.}$

Minimize $C - \sum_{k=1}^{m} Q_{imk}(X_{ik} + P_{ik}) \qquad (i - 1,2,\ldots\ldots n) - -\!\!\rightarrow (1)$

The HS parameters are specified in this step. These are Harmony Memory Size (HMS), or number of solution vectors in the harmony memory; Harmony Memory Considering Rate (HMCR); Pitch Adjusting Rate (PAR); and Number of Improvisations (NI) or stopping criterion. The Harmony Memory (HM) is a memory location where all the solution vectors (sets of decision variables) are stored. The parameters *HMCR* and *PAR* are used to improve the solution vector and these are defined in step 3.

Step2: Initialize the harmony memory

In this step, the HM matrix is filled with as many randomly generated solution vectors as the HMS:

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \dots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \dots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \dots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix} \begin{matrix} \Rightarrow & f(x^{(1)}) \\ \Rightarrow & f(x^{(2)}) \\ \vdots & \vdots \\ \Rightarrow & f(x^{(HMS-1)}) \\ \Rightarrow & f(x^{(HMS)}) \end{matrix} \quad (4)$$

Step3: Improvise a new harmony from the HM set

A new harmony vector, $x' = (x_1', x_2',..., x_n')$, is generated based on three rules, namely, random selection, memory consideration and pitch adjustment.

These rules are described as follows:

- *Random Selection:* When HS determines the value, x_i' for the new harmony, $x' = (x_1', x_2',..., x_n')$, it randomly picks any value from the total value range with a probability of (1-HMCR). Random selection is also used for previous memory initialization.

- *Memory Consideration:* When MBHS determines the value x_i', it randomly picks any value x_i^j from the HM with a probability of HMCR since j= {1,2,…, HMS}.

$$x'_i \leftarrow \begin{Bmatrix} x'_i \in \{x_i^1, x_i^2, \dots\dots, x_i^{HMS}\} & \textit{with probability HMCR} \\ x'_i \in X_i & \textit{with probability } (1\text{-}HMCR) \end{Bmatrix} (5)$$

- *Pitch Adjustment:* Every component of the new harmony vector, $x' = (x_1', x_2',..., x_n')$, 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:

$$x'_i \leftarrow \begin{Bmatrix} Yes & \textit{with probability PAR} \\ No & \textit{with probability } (1\text{-}PAR) \end{Bmatrix} \quad (6)$$

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

$$x'_i \leftarrow x'_i \pm bw \quad (7)$$

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.

Step 4: Updating HM

If the new harmony vector, $x' = (x_1', x_2',..., x'_n)$, is better than the worst harmony in the HM, from the viewpoint of the objective function value, the new harmony is entered in the HM and the existing worst harmony is omitted from the HM.

Step 5: Checking stopping criterion

Computation is terminated upon satisfying the maximum number of improvisations or maximum number of iterations, which is the stopping criterion. Otherwise, steps 3 and 4 are repeated. Finally the best harmony memory vector is selected and is considered to be the best solution to the problem under investigation.

2.1. Improved Harmony Search 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.

Small *PAR* values with large *bw* values can lead to poor performance of the algorithm and considerable increase in the number of iterations to find optimum solution. Small *bw* values in the final generations increase the fine tuning of solution vectors. Therefore, large *PAR* values with small *bw* values usually leads to improvement in obtaining the best solution in the final generation in which the algorithm converges to optimal solution vector [31]. The IMBHS has been developed by Madhavi et al. [32] and has been successfully applied to various benchmarking and standard engineering optimization problems like Minimization of Weight of the spring, welded beam design, Pressure vessel design (4&6 inequalities), constrained function, unconstrained functions (I&II) and disjoint feasible region. Numerical results proved that the improved algorithm can result in better solutions when compared to the conventional MBHS and other heuristic or deterministic methods. Figure 2 shows the complete working of the algorithm.

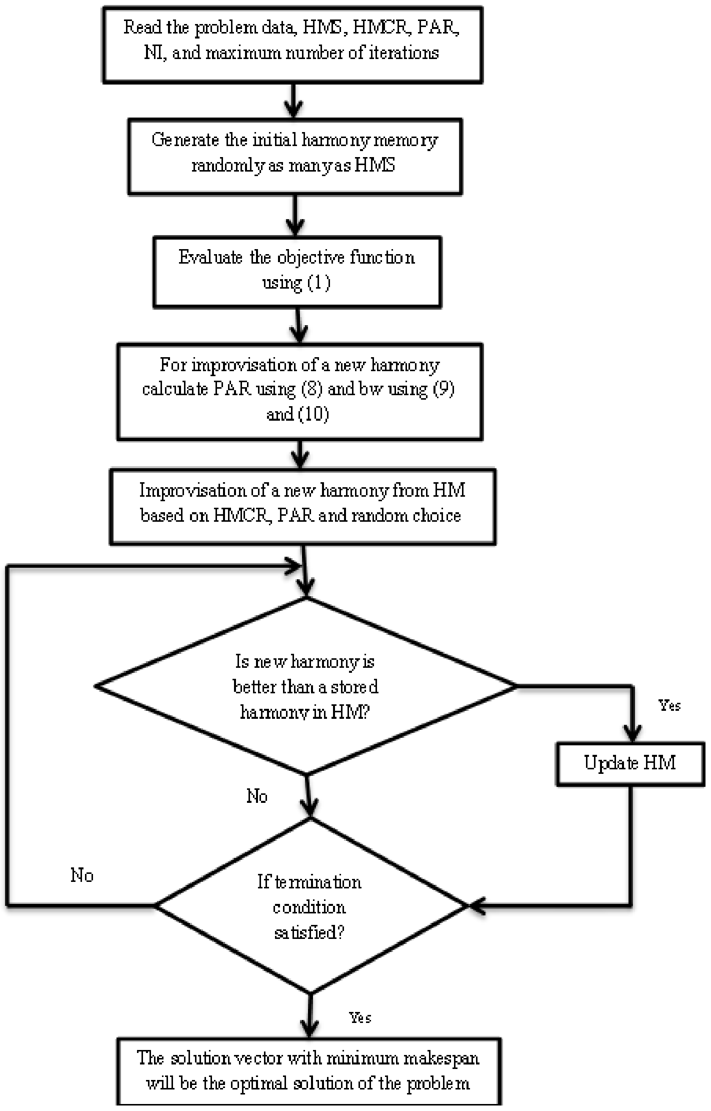
Read the problem data, HMS, HMCR, PAR, NI, and maximum number of iterations
Generate the initial harmony memory randomly as many as HMS
Evaluate the objective function using (1)
For improvisation of a new harmony calculate PAR using (8) and bw using (9) and (10)
Improvisation of a new harmony from HM based on HMCR, PAR and random choice
Is new harmony is better than a stored harmony in HM?
Yes
Update HM
No
No
If termination condition satisfied?
Yes
The solution vector with minimum makespan will be the optimal solution of the problem

Figure 2: Flowchart of IMBHS algorithm

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 as shown in Figure 3(a) and expressed as follows [13]:

$$PAR(gn) = PAR_{\min} + \left(\frac{PAR_{\max} - PAR_{\min}}{NI} \right) \times gn \qquad (8)$$

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.

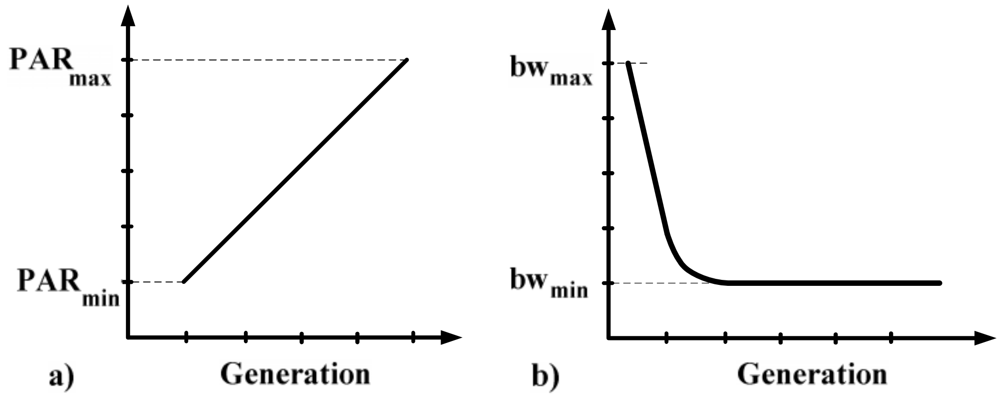

Figure 3: a) Variation of PAR versus generation number, b) Variation of *bw* versus generation number.

Bandwidth *(bw)* changes dynamically with generation number as shown in Figure 3(b) and defined as follows:

$$bw(gn) = bw_{\max}\ \exp(c.gn) \tag{9}$$

$$C = \frac{Ln\left(\dfrac{bw_{\max}}{bw_{\min}}\right)}{NI} \tag{10}$$

Where, *bw(gn)* is the bandwidth at each generation, bw_{min} and bw_{max} are the minimum and the maximum bandwidths respectively.

3. IMPLEMENTATION OF IMBHS FOR JSSP:

The solution to JSSP is a schedule of operation for jobs. In the present work IMBHSA is used to find the optimum schedule. A HM represents feasible schedule in this case. In the present work a direct approach "Operation based representation" is employed. Following is the brief description of the representation. The schedule is represented in the form of a string as shown in table 1. The representation encodes a schedule as a sequence of operations, and each character of the string stands for one operation. All operations for a job are represented by the same symbol, the job number and they are interpreted according to the order of their occurrence in the sequence in which they appear in the solution string. In this case the string is called vector solution of the HMS and the algorithm IMBHSA is used to evolve these vector solutions to discover potential schedules. For example, for a three-job-three-machine problem the representation would be as shown in table 1.

Table 1. Vector Solution representation of JSSP solution

3	2	1	2	1	3	3	1	2

3.1 Inputs for the problem

For solving JSSP using IMBHSA, the following inputs are required:

- Number of jobs (n)
- Number of Machines (m)

- Machine order for all the jobs (M_{ij})
- Processing Times of all the operations (P_{ij})
- Number of operations (O_{ij})
- Number of iterations to be carried out (n_{iter})
- Harmony Memory Size (depending on the problem size)
- Harmony Memory Consideration Rate (0.85)
- Pitch Adjusting Rate (PAR) (PAR_{min}=0.4, PAR_{max}=0.99)
- Band Width (BW) (BW_{min}=0.0001, BW_{max}=1)
- Number of Improvisations (NI=20)
- Maximum population allowable

3.2 Step wise Implementation of IMBHS for JSSP

This section gives a detailed explanation of implementation of the algorithm. The actual algorithm has been slightly modified for enhancing the performance. Improved Music Based Harmony Search Algorithm (IMBHSA) adopts process of HM for finding optimum solutions efficiently. In this case HMS is a potential solution and the algorithm helps HMS to evolve and generate better population thus giving rise to fitter solutions which represent competitive schedules.

3.2.1 Performance Evaluation

The initial population is made to reproduce depending on the fitness, the objective being minimizing makespan. In Scheduling literature, make-span is defined as the maximum completion time of all jobs, or the time taken to complete the last job on the last machine in the schedule-assuming that the processing of the first job began at time 0. Make-span is denoted by C_{max} and computed as $C_{max} = \max \{F_j\}$, where F_j is the flow time for job j(the total time taken by job j from the instant of its release to the shop to the time its processing by the last machine is over). Make span can be computed using (1).

3.2.2 Initialization the problem and algorithm parameters

Consider an optimization problem size of 6 machines and 6 jobs. This problem can be explained in the following steps by using IMBHS algorithms. Initially the HS parameters are defined according to the problem size.

Table 2. Parameters of IMBHS

HMS	10
HMCR	0.85
PAR_{min}	0.4
PAR_{max}	0.99
BW_{min}	0.0001
BW_{max}	1
NI	20
No.of iterations	50

3.2.3 Initialize the harmony Memory

In this step, the HM matrix is filled with as many randomly generated solution vectors as the HMS. For the above mentioned problem the order of the Harmony Memory Size is 36×36. In this HMS each row represents the one vector schedule. So initially we are generating 36 schedules randomly by following the problem constraints. Evaluate the objective function by using (1).

3.2.4 Improvise a new memory from the HM set

By using (8), (9) and (10) equations we are improvising the New Harmony memory from HM based on HMCR, PAR and Random Choice. Again we are we are evaluating the objective function with this New Harmony memory. If the solution is better than the stored harmony, update the HM with New Harmony memory otherwise stored HM is the updated HM for the next generation.

3.2.5. Checking the stopping criterion

The stopping criteria is one of the following

- Maximum number of iterations
- The population's worst and best fitness becomes equal

The cycle of steps explained is carried out until one of the stopping criteria is met.

4. PERFORMANCE OF IMBHS ON BENCH MARK PROBLEMS

IMBHS algorithm has been coded in Matlab and runs on Intel Core2Duo Processor with 4 GB RAM and CPU 2.00 Ghz. The bench marking problems from OR library, were solved by the proposed method. The results obtained were tabulated in table 3. In view of the results obtained by implementing IMBHS to solve JSSP, it appears that IMBHS is efficient. The algorithm has been improved by changing its solution coding method and hybridizing leading to fast convergence. The algorithm's performance has been compared to that of with the help of some bench-mark problems and has been found to be superior to the latter. It is concluded that the application of IMBHS to for solving JSSP is a new area to be explored for competitive solutions**.**

4.1. Results and Discussions

Table 3. Results of some Bench Mark Problems

S.NO	PROBLEM	Best Known Solution	MBHS(100000)		IMBHS(10000)	
			MAKESPAN	CPU	MAKESPAN	CPU
1	FT06 (6*6)	55	55	62.21	55	66.34
2	FT10 (10*10)	655	655	898.88	655	818.76
3	FT20 (20*5)	1119	1154	1367.23	1119	1163.21
4	LA01 (10*5)	666	666	164.04	666	154.46
5	LA06 (15*5)	926	926	554.11	926	504.20
6	LA11 (20*5)	1222	1245	2836.8	1222	1347.8
7	LA16 (10*10)	945	1021	2891.5	945	1342.6
8	LA21 (15*10)	1046	1066	2234.9	1046	1126.7
9	LA26 (20*10)	1218	1220	2198.5	1218	1173.9
10	LA31 (30*10)	1784	1798	2289.6	1784	1176.5
11	LA40 (15*15)	1222	1222	2517.6	1222	1259.2
12	SWV1(20*10)	1219	1220	2134.8	1219	1654.6
13	SWV6(20*15)	1229	1229	2289.4	1229	1548.3
14	SWV11(50*10)	2808	2808	2801.7	2808	1835.8
15	ABZ5(10*10)	1234	1234	2145.8	1234	1178.8
16	ABZ7(20*15)	556	556	2201.3	556	1341.5
17	YN1(20*20)	694	734	2789.5	694	2018.9
18	TA01(15*15)	977	977	2005.8	977	1867.5
19	TA11(20*15)	1139	1139	2126.7	1139	1899.1

20	TA21(20*20)	1217	1217	2276.4	1217	1959.8
21	CAR1(11*5)	7738	7758	2345.5	7738	2634.9
22	CAR2(13*4)	7166	7265	2143.7	7166	2376.9
23	CAR3(12*5)	7312	7369	2276.5	7312	2176.8
24	CAR4(14*4)	8003	8063	2456.8	8003	2265.9
25	CAR5(10*6)	7702	7708	2378.6	7702	2578.5
26	CAR6(8*9)	8313	8326	2156.8	8313	2376.8
27	CAR7(7*7)	6558	6578	2245.6	6558	2716.4
28	CAR8(8*8)	8264	8288	2278.1	8264	2673.9

Table 4. Comparison of Results of MBHS, IMBHS and HBSA with Bench Mark Problems

S.NO	PROBLEM	Best Known Solution	MBHS	IMBHS	HBSA
1	LA01 (10*5)	666	666	666	666
2	LA02(10*5)	655	674	655	668
3	LA03(10*5)	597	63	597	617
4	LA04(10*5)	590	598	590	604
5	LA05(10*5)	593	593	593	593
6	ABZ5(10*10)	1234	1234	1234	1234
7	ABZ6(10*10)	943	943	943	943
8	LA19(10*10)	842	844	842	842
9	LA20(10*10)	902	917	902	902
10	LA06(15*5)	926	926	926	926
11	LA07(15*5)	890	890	890	890
12	LA08(15*5)	863	876	863	863
13	LA09(15*5)	951	951	951	951
14	LA10(15*5)	958	976	958	958
15	LA21(15*10)	1046	1066	1046	1053
16	LA24(15*10)	935	945	935	935
17	LA25(15*10)	977	979	977	977
18	LA27(20*10)	1235	1239	1235	1269
19	LA29(20*10)	1157	1167	1157	1195
20	ABZ7(20*15)	556	556	556	668
21	ABZ8(20*15)	566	566	566	687
22	ABZ9(20*15)	563	563	563	707

It is to note that optimum solutions were found for all 120 problems by MBHS and IMBHS algorithms within 100000 iterations of run and 10000 iterations of run respectively. When we compared these two methods CPU time is more for MBHS method. But these two methods are giving best optimal solutions with maximum number of iterations. From Table 4, we can conclude that MBHS and IMBHS are giving best optimal solutions with Hybrid Method HBSA which is a combination of Bacterial Foraging Optimization (BFO) and Harmony Search (HS) proposed by Shivakumar B L and Amudha T.

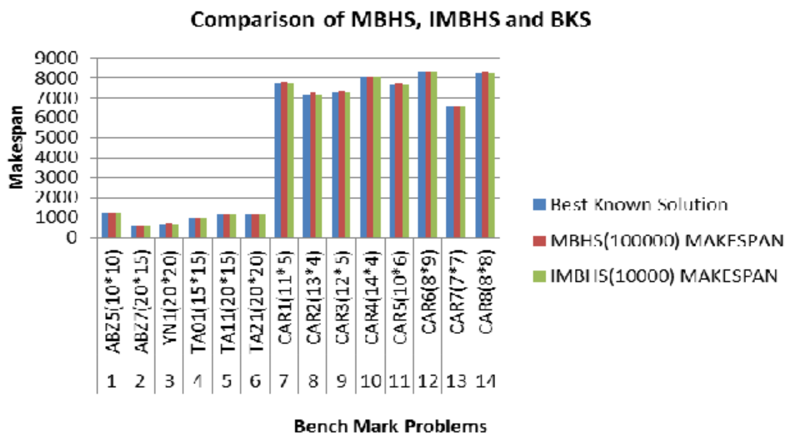

Figure 4: Comparison of MBHS, IMBHS and BKS for ABZ, YN, TA and CAR problems

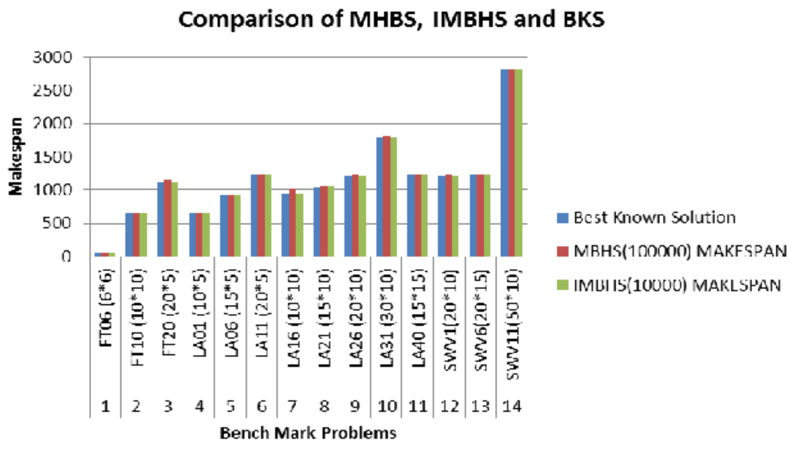

Figure 5. Comparison of MBHS, IMBHS and BKS for FT, LA and SWV problems

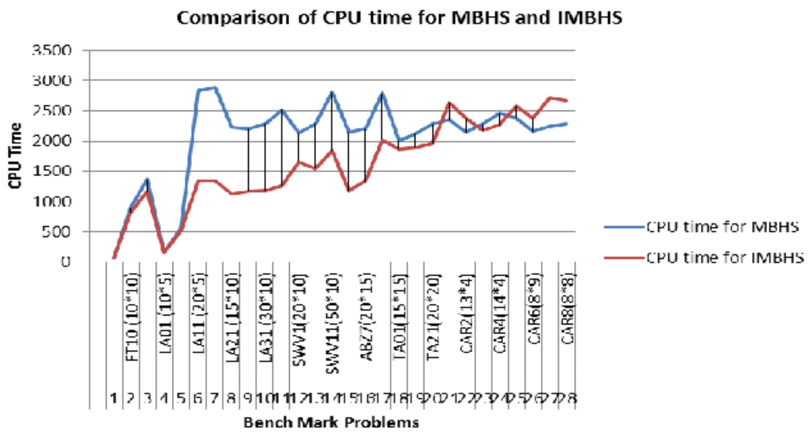

Figure 6. CPU time Comparison of MBHS, IMBHS for Bench Mark Problems

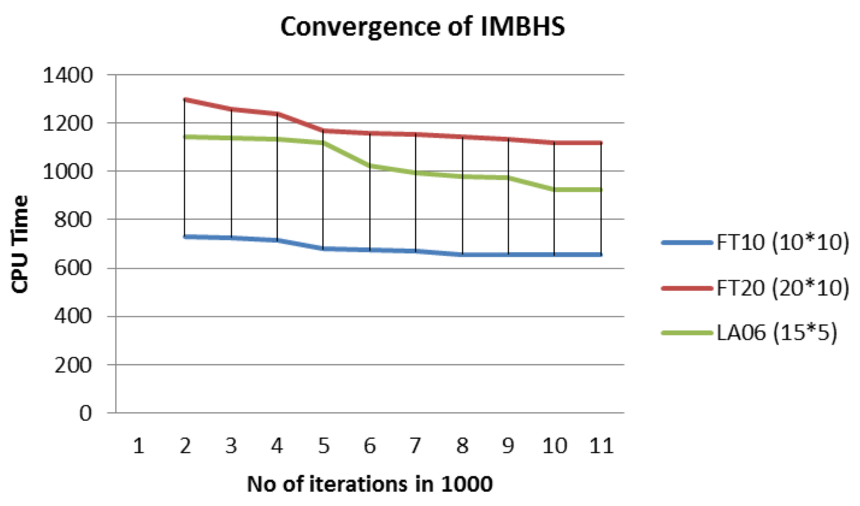

Figure 7. Convergence of IMBHS for 3 Bench Mark Problems

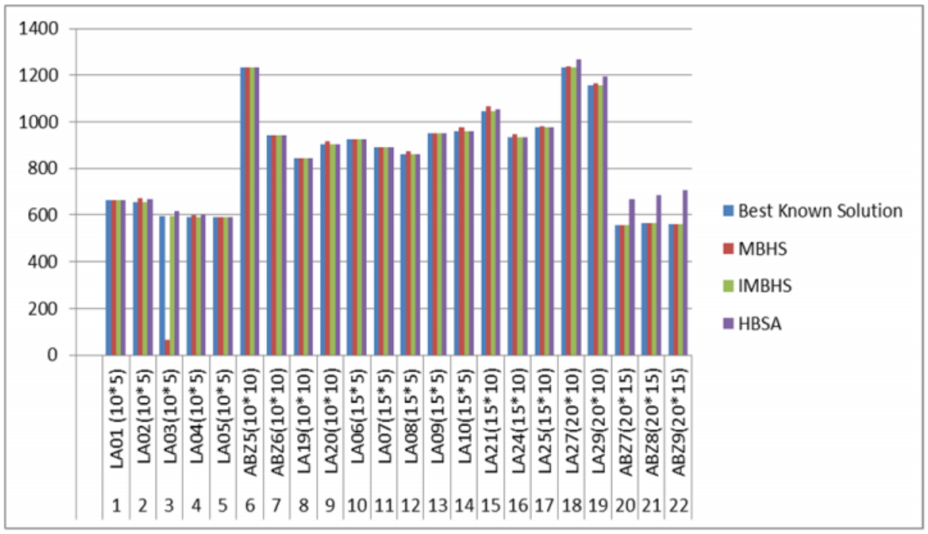

Figure 8. Comparison of MBHS, IMHS & HBSA with Best Known Solution

5. CONCLUSIONS

1. A thorough research survey indicate that Music Based algorithm as Metaheuristic algorithms has not been applied to JSSP and the authors have applied MBHS algorithms as most successful algorithms for JSSPs.
2. Job Shop Scheduling Problems (JSSPs) have been mapped in terms of Music based Harmonic Search and Improved Music Based Harmony Search algorithms and solved 120 bench mark problems though only 28 problems were reported here and also another 22 problems were reported in Table 2 is compared with HBSA method. It is found that IMBHS algorithm is efficient in generating solutions equal to BKS for all the bench mark instances. Hence IMBHS may be considered as an excellent tool for JSSPs.
3. MBHS has generated 50% of solutions equal to BKS and remaining solutions were almost equal. MBHS is performing equally with IMBHS on simple problems like FT06, whereas

IMBHS has generated 100% equal solutions. Hence IMBHS is superior to MBHS not only in searching good solutions but also time efficient. Hence IMBHS has emerged as a good alternative method for JSSPs.

REFERENCES

[1] Bruker P (1995), “Scheduling algorithms 2nd edn. Springer”, Berlin Heidelberg New York

[2] Garey M, et al (1976), “The complexity of flow shop and job shop scheduling”, Math Oper Res 1:117–129

[3] Erschler JF, Roubellat JP, Vernhes (1976), “Finding some essential characteristics of the feasible solutions for a scheduling problem”, Operations Research 24:774–783

[4] French S (1982), “Sequencing and scheduling: An introduction to the mathematics of the job shop”, Wiley, New York

[5] Carlier J, Pison E (1989) ,“An algorithm for solving the job shop problem”, Manage Sci 35:164–176

[6] Bruker P et al (1994) ,“A branch and bound algorithm for job shop problem”, Discrete Appl Math 49:107–127

[7] Blazewickz J (1996) ,“The job shop scheduling problem: Conventional and new solutions techniques”, Eur J Oper Res pp 931–30

[8] Pezzella F, Merelli E (2000) ,“A tabu search method guided by shifting bottleneck for the job shop scheduling problem”, Eur J Oper Res 120:297–310

[9] Balas E, Vazacopoulos A (1994) ,“Guided local search with shifting bottleneck for job shop scheduling”, Tech Rep Management Science Research Report MSRR-609, GSIA Carnegie Mellon University, Pittsburgh.

[10] Yang S, Wang D (2001) ,“A new adaptive neural network and heuristics hybrid approach for job shop scheduling”, Comput Oper Res 28:955–971

[11] Sudoku Puzzle: Geem, Z. W., “Harmony Search Algorithm for Solving Sudoku”, Lecture Notes in Artificial Intelligence, 2007.

[12] Tour Planning: Geem, Z. W., Tseng, C. -L., and Park, Y. “Harmony Search for Generalized Orienteering Problem: Best Touring in China”, Lecture Notes in Computer Science, 2005.

[13] Visual Tracking: J. Fourie, S. Mills and R. Green ,“Visual tracking using the harmony search algorithm”, Image and Vision Computing New Zealand, 2008. 23rd International Conference

[14] Visual Correspondence: J. Fourie, S. Mills and R. Green “Directed correspondence search: Finding feature correspondences in images using the Harmony Search algorithm”, Image and Vision Computing New Zealand, 23-25 Nov. 2009. 24rd International Conference.

[15] Design of radar codes: S. Gil-Lopez, J. Del Ser, S. Salcedo-Sanz, A. M. Perez-Bellido, J. M. Cabero and J. A. Portilla-Figueras, “A Hybrid Harmony Search Algorithm for the Spread Spectrum Radar Polyphase Codes Design Problem”, Expert Systems with Applications, Volume 39, Issue 12, pp. 11089-–11093, September 2012.

[16] Power and subcarrier allocation in OFDMA systems: J. Del Ser, M. N. Bilbao, S. Gil-Lopez, M. Matinmikko, S. Salcedo-Sanz, “Iterative Power and Subcarrier Allocation in Rate-Constrained OFDMA Downlink Systems based on Harmony Search Heuristics”, Elsevier Engineering Applications of Artificial Intelligence, Vol. 24, N. 5, pp. 748–756, August 2011.

[17] Efficient design of open Wifi networks: I. Landa-Torres, S. Gil-Lopez, J. Del Ser, S. Salcedo-Sanz, D. Manjarres, J. A. Portilla-Figueras, “Efficient Citywide Planning of Open WiFi Access Networks using Novel Grouping Harmony Search Heuristics”, accepted for its publication in Engineering Applications of Artificial Intelligence, May 2012.

[18] Single-objective localization: D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, R. Lopez-Valcarce, “A Novel Heuristic Approach for Distance- and Connectivity-based Multihop Node Localization in Wireless Sensor Networks”, Springer Soft Computing, accepted, June 2012.

[19] Bi-objective localization: D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, S. Salcedo-Sanz, R. Lopez-Valcarce, “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, in press, June 2012.

[20] Structural Design: Lee, K. S. and Geem, Z. W. “A New Structural Optimization Method Based on the Harmony Search Algorithm”, Computers & Structures, 2004.

[21] Water Network Design: Geem, Z. W. "Optimal Cost Design of Water Distribution Networks using Harmony Search", Engineering Optimization, 2006.

[22] Vehicle Routing: Geem, Z. W., Lee, K. S., and Park, Y. "Application of Harmony Search to Vehicle Routing", American Journal of Applied Sciences, 2005.

[23] Ground Water Modeling: Ayvaz, M. T. "Simultaneous Determination of Aquifer Parameters and Zone Structures with Fuzzy C-Means Clustering and Meta-Heuristic Harmony Search Algorithm", Advances in Water Resources, 2007.

[24] Soil Stability Analysis: Cheng, Y. M., Li, L., Lansivaara, T., Chi, S. C. and Sun, Y. J. "An Improved Harmony Search Minimization Algorithm Using Different Slip Surface Generation Methods for Slope Stability Analysis", Engineering Optimization, 2008.

[25] Satellite Heat Pipe Design: Geem, Z. W. and Hwangbo, H. "Application of Harmony Search to Multi-Objective Optimization for Satellite Heat Pipe Design", Proceedings of US-Korea Conference on Science, Technology, & Entrepreneurship (UKC 2006), CD-ROM, Teaneck, NJ, USA, Aug. 10-13 2006.

[26] Dam Scheduling: Geem, Z. W. "Optimal Scheduling of Multiple Dam System Using Harmony Search Algorithm", Lecture Notes in Computer Science, 2007.

[27] Ecological Conservation: Geem, Z. W. and Williams, J. C. "Ecological Optimization Using Harmony Search", Proceedings of American Conference on Applied Mathematics, Harvard University, Cambridge, MA, USA, March 24–26, 2008.

[28] Heat exchanger design: Fesanghary, M., Damangir, E. and Soleimani, I. "Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search", Applied Thermal Engineering, In press.

[29] Face milling: Zarei, O., Fesanghary, M., Farshi, B., Jalili Saffar, R. and Razfar, M.R. "Optimization of multi-pass face-milling via harmony search algorithm", Journal of Materials Processing Technology, In press.

[30] Multicast Routing: Forsat, R., Haghighat, M., Mahdavi, M., "Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing", Computer Communications, Elsevier

[31] K. Lee and Z. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer Methods in Applied Mechanics and Engineering* 194, pp.3902-3933,2005.

[32] M.Mahdavi, M.Fesanghary, E.Damangir, "An improved harmony search algorithm for solving optimization problems", *Applied Mathematics and Computation* 188 ,pp.1567–1579,2007.

Authors

Mrs. Hymavathi M passed B.Tech from G.E.C – 2008 and M.Tech from NIT Warnagal with Computer Integrated Manufacturing specialization in 2010. She is presently a Research Scholar at NIT Warangal. She had published 4 International Journal papers and 5 International Conference papers. Her Research interest includes Scheduling, Digital Manufacturing, Soft Computing Techniques, Naturally Inspired Algorithms and P.L.M. She was
Gold Medallist and Best Outgoing student at GEC – 2008.

Dr. C.S.P. Rao is Professor, Department of Mechanical Engineering, and National Institute of Technology (NIT) Warangal, India. He has over 25 years of teaching and research experience in the areas of CAD/CAM, Robotics, FMS, CIM, Soft Computing Techniques applications in Manufacturing and Simulation of Manufacturing Systems. He has published 250 papers in International Journals & Conferences and four popular text books CAD/CAM, Production Engineering, Engineering Drawing using AutoCAD and Robotics (in press). He was awarded A.P. Scientist & Engineer of the Year awards for 2008. He guided 16 Ph.D. Scholars & Presently guiding 10 Scholars. He executed several R&D / Sponsored Projects of MHRD, AICTE, DST & DRDO.