An Improved Bat Algorithm for the Hybrid Flowshop Scheduling to Minimize Total Job Completion Time

Minal Soni, Gaurav D. Sharma, Brij Kishore (Asst. Prof. AIET), Manish Sharma (Asst. Prof. AIET)
minalsoni782@gmail.com, dayitv89@gmail.com, akbrij@gmail.com, manishaiet17@gmail.com

Abstract:- In this paper, we present improved bat algorithm (BA) to solve hybrid flowshop scheduling (HFS) problem, which is a typi- cal NP-hard combinatorial optimization problem with strong engineering production back- grounds. To make algorithms applicable in the HFS problem, we use smallest position value (SPV) rule to associate particles continuous property to discrete job order, greedy method to compute this job order to complete HFS schedule and rank selection rule for particles local search. Computation has three major outcomes: total iteration required to solve the problem, total computation time needed and total job completion time (JCT). Simulation results based on a variety of instances demonstrate the effectiveness, efficiency, and robust- ness of the algorithms. Comparison with particle swarm optimization (PSO) algorithm de-picts that BA gives better results and stable outcomes.

Keywords:- *Hybrid flowshop scheduling (HFS) problem, Bat Algorithm (BA), Particle Swarm Optimization (PSO) Algorithm.*

*****

I. INTRODUCTION

Nowadays, It is very important to develop effective and efficient scheduling technologies and approaches, because production scheduling plays a key role in the manufacturing systems of an enterprise for maintaining a competitive position in the fast-changing market. Flow shop or hybrid flow shop is very familiar in process industry such as the iron and steel industry and the chemical industry. The HFS problem can be seen as a mixture of the flowshop scheduling and the parallel machine scheduling, so it is more complex and difficult to be solved. Even when there are only two stages in the HFS, it is also NP-hard [1]. In a HFS there is a set of production stages, where a stage includes some identical parallel machines (see Fig. 2). All jobs are processed in stages and pass through these stage in the same sequence. In each stage, a job can be processed on one and only one machine for only one time and assuming that infinite buffer capacities between two stages. The objective is to find a schedule so as to minimize a given performance measure such as makespan, total weighted completion time, maximum lateness, and so on.

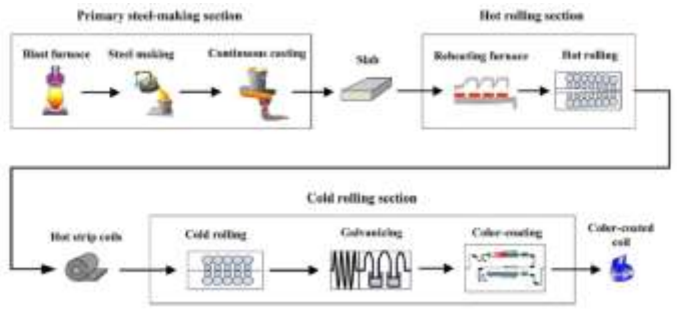

Fig. 1. Production Steps in the Iron & Steel Industry

A review of related research on HFS, including processing complexity, scheduling criterion and scheduling approaches, can be found in Linn and Zhang [2]. Branch and bound (B&B) and heuristics are the two most commonly employed approaches to HFS scheduling.

Many other algorithms have been proposed to solve HFS problem, like tabu search [3], six heuristics [4], artificial immune system [5], genetic algorithm(GA) [6], PSO algorithm [7] [8] etc. In the paper [9], compared the perfor- mance of several meta-heuristics such as simulated annealing, tabu search and genetic algorithm for the HFS with sequence- dependent and machine-dependent setup times. We choose BA due to the promising behavior of the algorithm. In the paper [10], comparison of BA, GA, PSO and other algorithms, BA performed well than other algorithms.

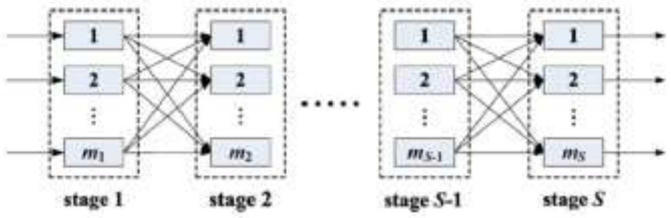

Fig. 2. Typical hybrid flowshop

Most of the researches on the HFS are focused on the criterion of minimizing makespan. However, in re- cent years the criterion of minimizing total weighted completion time has started to draw more attention from researchers because it is more relevant and meaningful for todays dynamic production environment and directly related to important manufacturing logistics. In this paper we consider the HFS problem to minimize P the total completion time (shows as fintess function = min(Cj)). This paper is organized as follows: The significance of BA as well as the improvement strategies incorporated with this algorithm is discussed in section II. Experiment and analysis are described in section III. Finally, section IV concludes the work.

II. ALGORITHMS FOR HFS

As one of the latest soft computing technique, the bat algorithm(BA) has been successfully applied in many problems such as the engineering design, classifications of gene expression data, ergonomic workplace problems etc. The experimental results showed that the BA have better performance as compared to GA, ACO, and PSO, whereas the hybrid BA(HBA) with varying neighbourhood search has outstanding performance. Due to the promising performance of BA, in this paper we develop an hybrid BA algorithm for the HFS problem.

A. Smallest Position Value Rule

Because of the consistent characters of the position qual- ities of particles in the BA and PSO, general result rep- resentation of HFS can't be specifically embraced in the algorithms. To begin with instate the particles with some arbitrary numbers, then change over this particles to discrete job request π as $\pi^t_i = \{ \pi^t_{i1}, \pi^t_{i2}, ..., \pi^t_{in} \}$. In which π^t_{ij} signify the employment organized in jth position. Table I portrays the π^t_{ij}

Dimention j	1	2	3	4	5	6	7	8
x_{ti}^j	0.54	-0.75	-1.02	-0.41	0.92	-1.2	0.23	0.12
Job π_{ij}^t	6	3	2	4	8	7	1	5

well. The jth position depend on upon the particles sorted expanding request.

B. Basic PSO Algorithm

Particle swarm optimization (PSO) algorithm is a compu- tation technique which mimics the behavior of flying birds and the means of exchanging information about their environment with one another. The population of PSO is called a swarm and each individual in the population of PSO is called a particle. The particles are controlled by the following equations:

$$V_{id}(K+1) = \omega V_{id}(k)+c_1r_1((p_{id}(k) - x_{id}(k)) + c_2r_2((p_{gd}(k) - x_{id}(k)) \quad (1)$$

$$x_{id}(K + 1) = x_{id}(k) + V_{id}(k + 1) \quad (2)$$

where i=1,2,...,m; m is the number of particles in swarm and d=1,2,...,D; D is the dimension of the particle; pid is the best position found by that particle so far; pgd is the best position found by any particle so far; v is the velocity of particle in a single dimension; x is the position of particle in a single dimension; k is the iteration number; ! is the inertial constants; c1 , c2 are acceleration constants; r1 , r2 are random numbers evenly distributed between [0,1].

C. Greedy Method

Take π as the expanding request of these n jobs in the first stage and build the relating complete schedule of HFS utilizing an greedy technique. That is, in PSO, the result is spoken to as the occupation stage rather than a complete complex HFS plan. Through this strategy, the result representation gets simple and the algorithms can then be pertinent and particle update equation, but PSO algorithm need to store previous iteration data for calculating particle best position. Thus PSO takes more space and time compare to BA.

in HFS. Algorithm- 1 depicts the greedy method step by step.

Algorithm 1: Greedy Method

```
1: Set j=1
2: while(j≤S)do
3:    Take $\pi^t_i$ as the increasing order of these n jobs in the first stage.
4:    Set k=Mj,and assign the first k jobs in $\pi^t_i$ on the Mj machines.
5:    Calculate the completion time of each job and update the first available machine in stage j and set k = k + 1
6:    while (k ≤ n ) do
7:        assign $\pi^t_{ik}$ on first available machine in stage j.
8:        Calculate the completion time of each job and reupdate the first available machine in stage j.
9:        Set k = k + 1
10: Set j = j + 1
```

Since this development strategy is a sort of greedy heuristic, the got effect is just a close-ideal schedule. With the complete calendar, the destination quality of this timetable could be then effectively decided.

D. Bat Algorithm

Bat algorithm has been developed by Xin-She Yang in 2010 [11]. The algorithm exploits the so called echolocation of bats. Bats use sonar echoes to detect and avoid obstacles. It is generally known, that sound pulses are transformed to frequency which reflects from obstacle. Bats can use time delay from emission to reflection and use it for navigation. They typically emit short loud, sound impulses. The pulse rate is usually defined as 10 to 20 times per second. After hitting and reflecting, bats transform their own pulse to useful information to gauge how far away the prey is. Bats are using wavelengths, that vary from range [0.7,17] mm or inbound frequencies [20,500] kHz. By implementation, pulse frequency and rate has to be defined. Pulse rate can be simply determined from range 0 to 1, where 0 means there is no emission and by 1, bats are emitting maximum. Algorithm- 2 illustrate the basic bat algorithm.

Initialization of the bat population is performed randomly. Generating the new solutions is performed by moving virtual bats according the following equations:

$$Qi = Q_{min} + (Q_{max} - Q_{min})\beta, \quad (3)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{best})Qi, \quad (4)$$

$$xt = x_i^{t-1} + v_i^t \quad (5)$$

$$x_t = x* + \varepsilon A_i^t[2\beta - 1] \quad (6)$$

where x_{best} is the global best position among all bat

particles, 2 [0, 1] is a random number, Qi is the frequency $0 \leq Q_{min} \leq Q_{max} \leq 1$. The rate of pulse emission r_i increases and the loudness A_i decreases. Both characteristics imitate natural bats, where the rate of pulse emission increases and the loudness decreases when a bat finds a prey. Mathematically, these characteristics are

captured with following equations:

$$A_i^t = \alpha A_i^{t-1}; \; r_i^t = r_i^0[1 - exp(-\gamma\varepsilon)] \quad (7)$$

Algorithm 2 Bat Algorithm

```
1: Data:Initialize bat population, number of iterations, and ∀,bat: initialize position vector $x_i$, frequency vector fi, loudness li, rate of emission ri
2: Result: Global best position, fitness value
3: While(number of iterations) do
4:     ∀, bat: Calculate fitness value and rank in increasing order
5:     Generate new solutions by updating velocities and bat particles using equations (1) to (3).
6:     local search to find if any better WS exist for a task
7:   If(rand ∈ (0,1) > $r_i$) then
8:       Select a solution among the best solutions
9:       Generate a local solution around the best solution using equation (4).
10:      Evaluate fitness of the updated bat position.
11:    If(rand ∈ (0,1) < Ai) and ($fitness_{current}$ < $fitness_{updated}$) then
         Bat position = updated bat position
12:      Decrease Ai, increase ri using equation (5).
```

α, β, γ are constants. Actually, the α parameter plays a similar role as the cooling factor in simulated annealing algorithm that controls the convergence rate of this algorithm.

E. Second Rank Selection Rule

Most of the algorithm in soft computing are only refer to the global best. Some particle could be near to the second global best toward to the fitness threshold value, so to keep this randomness of the particle we focus on the second

global best that has rank-2. This algorithm is used for the particles local search. In this algorithm flipconstant is used to determine that positional update require for particle. flipconstant is similar to the mutation behavior in genetic algorithm (GA). We use value of flipconstant as 0.5, it could vary on different problem. Detail of second rank selection rule(SRS rule) is available in algorithm- 3.

Algorithm 3 Second Rank Selection Rule

```
1:Initialize $flip_{constant}$, set i = 1, n = total number of particles.
2: While( i ≤ n) do
3:   Set α = random number (0,1)
4:    If(α ≥ $flip_{constant}$) then
5:      Set β = random number (0,1)
6:      update velocity
          $v_i^t = v_i^{t-1} + (gBest_2 - x_i)* \beta$
        If ($gBest_2 \geq x_i$) then
          $x_i^t = x_i^{t-1} + v_i^t$
        Else
          $x_i^t = x_i^{t-1} - v_i^t$
```

F. Position Search Algorithm

Algorithm- 5 illustrate the position search algorithm(PS). Objective of this algorithm to minimize search domain of particles in BA and compute the effective job order to achieve efficient job completion time. This algorithm select the maximum occurrence of job for each index and set the initial job order for SPV rule. For example consider that k = 4 and n = 10, where k is the total job lists, and n is total number of jobs.

e.g. [2, 4, 1, 5, 7, ..., nth], [4, 2, 1, 6, 7, ..., nth], [2,3,1,5,9,...,nth],[2,9,8,6,3,...,nth]

For index = 1, 2 repeated three times; index = 2, no job is repeated so leave Empty; index = 3, 1 is repeated three times; index = 4, 5 is repeated twice and 6 is repeated twice, then select last one i.e. 6; index = 5, 7 repeated twice and so on. Then finally job list becomes [2, , 1, 6, 7, ..., nth]. 0 0 denotes the Empty job index. For empty space apply the bat particles to achieve efficient job completion time.

SPV rule is applied to mapping between the particles and jobs, greedy method is used to compute the complete HFS

Algorithm 4 Position Search Algorithm

```
1: Find the k job orders list, according to least job completion time. Find occurrence of job for each
2: index and select maximum.
3: if all jobs are distinct then
4:     leave Empty that index
5: if one job repeated in the list with different index then
6:     if occurrence are same then
7:         take first one and remove others as Empty
8:     else
9:         take maximum occur and remove others as Empty
10: Set the job order direct to SPV rule and for empty job index re-apply the bat algorithm.
```

schedule, SRS rule is applied for particle local search and particle update & find the job schedule BA and PSO applied. PS algorithm is used to improve the job completion time in BA and PSO.

G. Complete Solution Representation SPV rule is applied for the discrete job order, greedy method used for the completion of job schedule, BA is applied for find out the new solution and for local search in particles we used SRS rule. Algorithm- 4 illustrate the complete solution representation with our proposed improved BA. In case of PSO algorithm just need to change the BA velocity and particle update equation, but PSO algorithm need to store previous iteration data for calculating particle best position. Thus PSO takes more space and time compare to BA.

ALGORITHMS

Algorithm 5 Improved Bat Algorithm

```
1: Initialize bat population, ∀ ,bat initialize position vector xi, frequency vector fi, loudness li, rate of emission ri.
2: Set iteration t : 1 number of iterations T, x∗  Global best position: π1t , f itnessrepeatation : 0 and fitnessthreshold.
3: while(t ≤ T) do
4:     Apply SPV rule to ∀ , bat particles.
5:     Apply greedy method to given discrete job or-
       der from SPV rule. Calculate fitness value and update x∗ Global best position.
6:     Generate new solutions by updating velocities and bat particles using equations (1) to (3).
7:     For local search apply SRS Rule to update bat particles.
8:     if (rand ∈ (0,1) < Ai) and (fitnesscurrent < fitnessupdated) then
9:         Bat position = updated bat position
10:        Decrease Ai, increase ri using equation (5).
11:        if (fitnesscurrent = fitnessupdated) then
12:            fitnessrepeatation = fitnessrepeatation + 1
13:        else
14:            fitnessrepeatation = 0
15:        if (fitnessrepeatation ≥ fitnessthreshold) then
16:            Break While loop
17:     Set t = t + 1
18: Calculate Computation Time required for t.
```

III.COMPUTATIONAL EXPERIMENTS & RESULTS

Here bat algorithm(BA), PSO algorithm(PSO), position search algorithm with BA(BA-PS) and PSO(PSO-PS) are successfully applied to HFS problem on 11 different type of datasets. Dataset is taken from the ”School of Business and Engineering Vaud, Switzerland” [12]. This experiment is repeated many times, calculated the average value of the outcomes and evaluated the below results. I have considered three parameters: job completion time (JCT), total iterations and total computation time(in seconds). Results are compared between BA and PSO Algorithm. This experiment is done on Python 2.7, Mac OS X 10.12.6, Intel i5 processor, 8 GB DDR3 RAM. Different platform may give different results.

$$\text{Deviation\%} = (f_{average} - f_{minimum})*100 / f_{minimum}$$

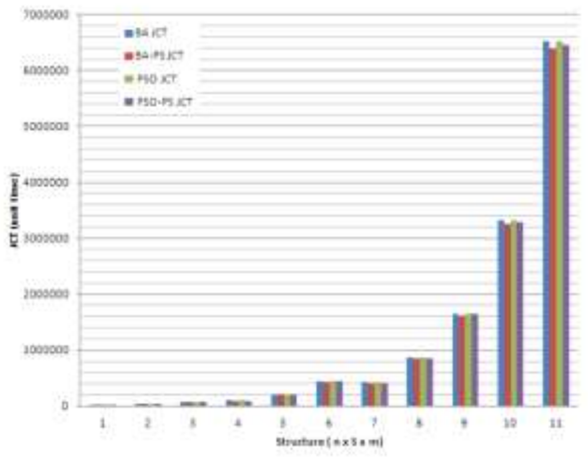

Fig. 3. Average Job Completion Time for BA and PSO

Fig. 3 depicts the average job completion time(JCT) for BA, BA-PS, PSO and PSO-PS algorithms. X axis shows the 11 different structures of the datasets, which is in the form of (nxSxm)wherenisthetotalnumberofjobs,Sistotal stages and m is the machine in each stage. Graph shows that PS algorithm play a vital role to find JCT, thus BA-PS and PSO-PS always gives better results compare to BA and PSO. BA and PSO overall gives the same results. BA-PS fulfill the objective to minimize the job completion time very well.

TABLE II. Average Job Completion Time & Deviation %

		BA		PSO		BA-PS		PSO-PS	
S.No.	Structure	JCT	deviation %	JCT	deviation %	JCT	deviation %	JCT	deviation %
1	20 x 5 x 3	18275.7	1.740801	18323.25	2.079387	17876.8	1.624694	18105.45	2.458548
2	20 x 10 x 3	36564.05	1.218165	36561.15	1.810448	35727.7	1.432871	36125.85	2.365617
3	20 x 20 x 3	73083	1.335275	72801.05	1.157528	71669.6	2.052742	72132	1.132859
4	50 x 5 x 3	99141.4	1.31534	99090.75	1.371611	97233.95	2.428078	98303.2	3.554446
5	50 x 10 x 3	210577.7	1.179933	209951.75	1.482836	205503.85	2.028543	208429.55	1.617929
6	50 x 20 x 3	443428.75	0.644306	443654.45	0.747211	434328.45	0.473871	439281.7	1.568728
7	100 x 5 x 3	423458.9	1.276398	423184	1.47543	415088.75	0.523032	419765.55	1.27327
8	100 x 10 x 3	863688.1	0.767122	864925.15	1.300762	847497.1	1.05697	854788.25	1.424597
9	100 x 20 x 3	1653101.2	0.693433	1655018.45	0.636615	1620756.2	0.849051	1639215.75	1.03554
10	200 x 10 x 3	3324331.4	0.568817	3326996.25	0.48707	3260093	0.471927	3294100.4	1.113516
11	200 x 20 x 3	6521083.75	0.471239	6523901.6	0.468417	6391755.55	0.165558	6460945.2	1.219758

Table II depicts the average value of job completion time(JCT) and their deviation percentage for the algorithms. Deviation percentage is calculated by the equation 8. Table shows that deviation percentage of BA-PS and BA is almost same, PSO and PSO-PS deviation percentage is overall high than BA and BA-PS. Deviation percentage shows the average result varies from the minimum(lower bound). It does mean BA-PS gives better and stable results than other algorithms on the basis of JCT.

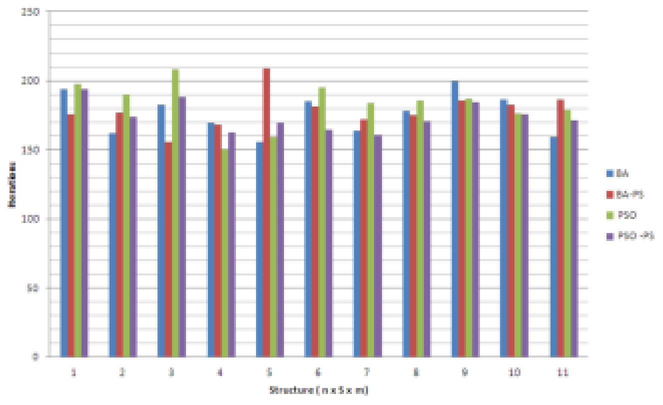

Fig. 4. Average iterations & % deviation

Fig. 4 depicts the average iteration require for all algo- rithms applied to HFS. X axis shows the 11 different structures of the datasets, which is in the form of (n x S x m) where n is the total number of jobs, S is total stages and m is the machine in each stage. Graph shows that overall the BA require less iteration to solve the problem compare to PSO algorithm. BA-PS and PSO-PS takes more iterations than BA and PSO. Iterations are whole number but after averaging of the iterations it is assume as real number.

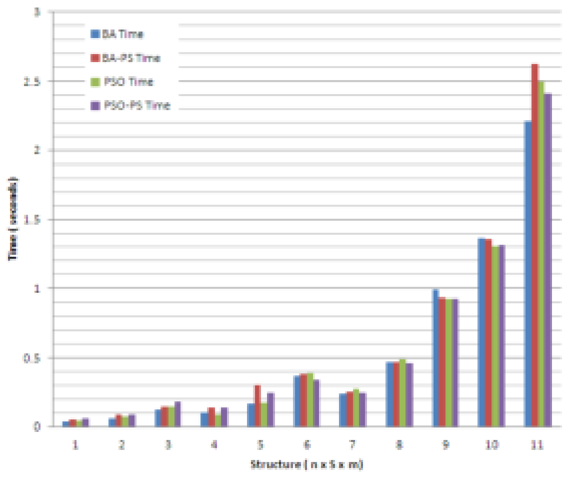

Fig. 5. Average job completion time & % deviation

TABLE III. Average Iterations & Deviation %

		BA		PSO		BA-PS		PSO-PS	
S.No.	Structure	Iterations	Deviation %	Iterations	Deviation %	Iterations	Deviation %	Iterations	Deviation %
1	20 x 5 x 3	193.9	74.684685	197.7	81.376147	175.95	74.207921	193.7	89.901961
2	20 x 10 x 3	161.8	51.214953	190.05	77.616822	177.1	75.346535	173.95	70.539216
3	20 x 20 x 3	182.45	78.872549	208.2	94.579439	155.6	44.074074	188.2	77.54717
4	50 x 5 x 3	169.45	62.932692	150.35	48.861386	167.85	55.416667	162.6	56.346154
5	50 x 10 x 3	155.8	54.257426	159.35	53.221154	209.15	90.136364	169.15	61.095238
6	50 x 20 x 3	185.25	83.415842	195.25	93.316832	181.05	77.5	164.15	60.931373
7	100 x 5 x 3	164	62.376238	183.6	80	171.9	50.789474	160.8	51.698113
8	100 x 10 x 3	178.15	68.066038	185.85	75.330189	175.35	73.613861	170.35	67.009804
9	100 x 20 x 3	199.85	86.775701	187.2	62.782609	186	80.582524	184.2	70.555556
10	200 x 10 x 3	186.3	72.5	176.5	69.711538	180.45	78.872549	175.4	71.960784
11	200 x 20 x 3	159.1	57.524752	179.1	64.311927	176.45	72.638889	171.1	67.745098

Table III depicts the average value of iterations and their deviation percentage for both of the algorithms. Table also shows that BA deviation percentage is less than the PSO deviation, BA-PS and PSO-PS deviation percentage are almost same. Which means that BA is almost stable compare to other algorithms in term of iteration requires.

Fig. 5 depicts the average computation time in seconds require for all algorithms to solve the HFS problem. X axis shows the 11 different structures of the datasets, which is in the form of (n x S x m) where n is the total number of jobs, S is total stages and m is the machine in each stage. Graph shows that overall the BA require less computation time to solve the problem compare to PSO algorithm. BA-PS and PSO-PS almost require almost same time.

TABLE IV. Average Computation Time & Deviation %

		BA		PSO		BA-PS		PSO-PS	
S.No.	**Structure**	**Time**	**Deviation %**	**Time**	**Deviation %**	**Time**	**Deviation %**	**Time**	**Deviation %**
1	20 x 5 x 3	0.04145	80.217324	0.0434	80.832125	0.051005	74.139139	0.057873	91.000127
2	20 x 10 x 3	0.06085	52.125146	0.0728	82.000155	0.090041	74.804309	0.08984	71.057569
3	20 x 20 x 3	0.127	81.428691	0.14625	92.434317	0.147309	44.405866	0.179426	77.650144
4	50 x 5 x 3	0.10055	62.177318	0.0917	47.903717	0.136754	55.631959	0.135491	59.702601
5	50 x 10 x 3	0.1649	54.11197	0.171	52.678601	0.300374	89.426672	0.246188	60.361894
6	50 x 20 x 3	0.36815	82.252561	0.3899	92.068903	0.383043	85.906867	0.340269	63.543622
7	100 x 5 x 3	0.23905	61.520257	0.27235	78.006409	0.255581	49.733818	0.243309	50.562407
8	100 x 10 x 3	0.4674	66.928598	0.4924	74.609877	0.46763	74.060129	0.458692	66.966869
9	100 x 20 x 3	0.98815	85.742459	0.92935	62.190239	0.931877	80.205052	0.92622	69.197473
10	200 x 10 x 3	1.3676	72.025142	1.30425	69.38312	1.317794	77.824065	1.314818	72.195812
11	200 x 20 x 3	2.20795	56.814634	2.49405	64.515191	2.323677	72.033088	2.412435	67.617548

Table IV depicts the average value of computation time and their deviation percentage for the algorithms. Table also shows that BA deviation percentage is less than the PSO. Which means that BA is more stable than PSO algorithm in term of computation time. Deviation percentage of BA-PS and PSO-PS are almost equal.

TABLE V. Comparative Analysis of Algorithms

	JCT	Iterations	Time
BA	Average	Very Good	Very Good
PSO	Average	Good	Good
BA-PS	Very Good	Average	Average
PSO-PS	Good	Average	Average

Other parameter outcomes could be considered on almost equal time, because difference among the time taken by the algorithm is in milliseconds which is negligible.

Comparative analysis of BA, PSO, BA-PS and PSO- PS is discussed in table V.

IV. CONCLUSION

In this paper, we encounter the HFS problem with the criteria of minimising the total completion time. An improved BA is proposed to solve this problem. For discrete job order and continuous property of the algorithms, a mapping rule using SPV rule. A greedy constructive method is developed for construct the cor- responding complete job schedule. SRS rule is applied to local search for the particles. Though the SPV and greedy method HFS problem can easily solved by any continuous soft computing algorithm. Overall the computation results show that BA has less time and space complexity, higher efficient, stable and robust than PSO algorithm. Position Search algorithm(PS) is used to reduce the particle search domain and to achieve efficient job completion time. With BA and PSO it gives better result than simple BA and PSO. To minimize job completion time, bat algorithm with position search algorithm(BA-PS) gives better results. Further research may be focused on the job-shop scheduling, open shop scheduling, for more complex HFS as each stages has two types of machines and machines eligibility criteria can be taken for development greedy method.

ACKNOWLEDGMENT

This research was supported by Gaurav Sharma & Brij Kishore(Asst. Prof. AIET) provided expertise that greatly assisted the research.I am grateful to the members for their patience and support in overcoming numerous obstacles I have been facing through my research.I have to express out appreciation to the Manish Sharma (Asst. Prof. AIET)

& Manish Sharma (Asst. Prof. AIET) for sharing their pearls of wisdom with me during the course of this research, although any errors are my own and should not tarnish the reputations of these esteemed professionals.

REFERENCES

[1] Brah, S. A., and John L Hunsucker. 1991. Eu- ropean Journal of Operational Research 51 (1):88– 99.

[2] Fister Jr , I., Dušan Fister , and Xin-She Yang. 2013. arXiv preprint arXiv:1303.6310 .

[3] Gary, M. R., and David S Johnson. 1979.

[4] Kennedy, J., and RC Eberhart. 1995. p. 1942– 1948. In: Procs of the International Conference on Neural Network .

[5] Khan, K., and Ashok Sahai. 2012. Interna- tional Journal of Intelligent Systems and Applica- tions (IJISA) 4 (7):23.

[6] Lenin, K., B Ravindranath Reddy, and M Surya Kalavathi. a. . title .

[7] Li, B.-B., Ling Wang, and Bo Liu. 2008. Sys- tems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 38 (4):818–831.

[8] Liu, B., Juan-Juan Xu, Bin Qian, Jian-Rong Wang, and Yan-Bin Chu. 2013. p. 60–64. In: Memetic Computing (MC), 2013 IEEE Workshop on . IEEE.

[9] Nejati, M., Iraj Mahdavi, Reza Hassan- zadeh, Nezam Mahdavi-Amiri, and MohamadSailm Mo jarad. 2013. The International Journal of Advanced Manufacturing Technology :1– 14.

[10] Pant, M., T Radha, and Ved Pal Singh. 2007. p. 3294–3299. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on . IEEE.

[11] Qunxian, C. 2010. p. 296–299. In: Mea- suring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on vol. 2. IEEE.

[12] Tang, L., and Xianpeng Wang. 2010. Con- trol Systems Technology, IEEE Transactions on 18 (6):1303–1314.

[13] Tang, L., Hua Xuan, and Jiyin Liu. 2006. Computers & Operations Research 33 (11):3344– 3359.

[14] Vignier, A., J-C Billaut, C Proust, and V Tkindt. 1996. p. 2934–2941. In: Systems, Man, and Cybernetics, 1996., IEEE International Confer- ence on vol. 4. IEEE.

[15] Yang, X.-S. 2010. In: Nature inspired cooper- ative strategies for optimization (NICSO 2010) p. 65–74. Springer.

[16] Yang, X.-S., and Xingshi He. a. . International Journal of Bio-Inspired Computation 5 (3):141–149.