Journal of Theoretical and Applied Information Technology
30th April 2018. Vol.96. No 8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN EFFICIENT ARTIFICIAL FISH SWARM ALGORITHM WITH HARMONY SEARCH FOR SCHEDULING IN FLEXIBLE JOB-SHOP PROBLEM

[1]ISHRAQ F. FAEQ, [2]MEHDI G. DUAIMI, [3]AHMED T. SADIQ AL-OBAIDI

[1,2]Dept. of Computer Science, College of Sciences, University of Baghdad, Baghdad, Iraq

[3]Dept. of Computer Science, University of Technology, Baghdad, Iraq.

E-mail: [1]hiishraqhifouad@gmail.com, [2]mehdi_duaimi@ymail.com, [3]drahmaed_tark@yahoo.com

ABSTRACT

Flexible job-shop scheduling problem (FJSP) is one of the instances in flexible manufacturing systems. It is considered as a very complex to control. Hence generating a control system for this problem domain is difficult. FJSP inherits the job-shop scheduling problem characteristics. It has an additional decision level to the sequencing one which allows the operations to be processed on any machine among a set of available machines at a facility. In this article, we present Artificial Fish Swarm Algorithm with Harmony Search for solving the flexible job shop scheduling problem. It is based on the new harmony improvised from results obtained by artificial fish swarm algorithm. This improvised solution is sent to comparison to an overall best solution. When it is the better one, it replaces with the artificial fish swarm solution from which this solution was improvised. Meanwhile the best improvised solutions are carried over to the Harmony Memory. The objective is to minimize a total completion time (makespan) and to make the proposed approach as a portion of the expert and the intelligent scheduling system for remanufacturing decision support. Harmony search algorithm has demonstrated to be efficient, simple and strong optimization algorithm. The ability of exploration in any optimization algorithm is one of the key points. The obtained optimization results show that the proposed algorithm provides better exploitation ability and enjoys fast convergence to the optimum solution. As well, comparisons with the original artificial fish swarm algorithm demonstrate improved efficiency.

Keywords: *Artificial Fish Swarm Algorithm; Harmony Search; Makespan; Flexible Job-Shop Scheduling Problem.*

1. INTRODUCTION

Scheduling is a process of decision making and an important tool in both the manufacturing and service industries. It deals with the distribution of operations on machines (i.e., the operations sequence on machines) in a way that performance goal which called makespan can be minimized. Scheduling of activities should be efficient enough to use the available resources in an effective manner. To address these issues, shop scheduling has motivated the researchers in both academia and industry. A variety of scheduling problems have been acknowledged from real-world manufacturing environments.

In this paper, our interesting is in the flexible job shop scheduling problem (FJSP) which is a generalization of the conventional job shop problem. Each operation can be performed by a given machine that is selected from a limited subset of candidate machines. The FJSP is more sophisticated and complex than the traditional JSP since it appends a new decision level to the sequencing one, i.e., the machine assignment that involves the selection of one machine among the available ones for each operation. The target is to find an allocation for each operation and to define the sequence of operations on each machine to minimize the maximum competition time called the makespan [1].

For its robustly NP-hard nature, many effective heuristics and meta-heuristics methods are improved and developed to get nearby optimal solutions which satisfy the qualifications and minimize or maximize the objective function ([1]; [2]; [3]; [4]; [5]). Among these methods, we have Artificial Fish Swarm Algorithm [6] Harmony Search Algorithm [7].

This paper investigates an Artificial Fish Swarm Algorithm with Harmony Search (AFSA-HS) for Flexible Job Shop Scheduling problem (FJSP). Most species of animals show social behaviors. In some

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

species, this is the top member of the group which leads all members of that group. For example, this behavior pattern is very clear in lions, monkeys, and deer. However, there are other classes of animals which live together in groups but they don't have a leader. In this type of animals, each member has a self-organized behavior which enables it to move around its environment and response to its natural needs with no need to leaders like birds, fishes and sheep drives. This type of animals has no knowledge about their group and the environment. Instead, through exchanging data with their adjacent members they can move in the environment. This simple interaction among particles makes group behavior more sophisticated as if we are looking for a particle in a wide environment.

Artificial fish-swarm algorithm (AFSA) is a relatively modern addition to the scope of natural computing. It has elements inspired by the social behaviors of naturalist swarms and linked with evolutionary computation. AFSA has found as an application widespread in complex optimization domains, and presently a prime research topic, offering an alternative to the more established evolutionary computation techniques that may be applied in many of the same domains.

The harmony search algorithm (HS) [8] is one of the population-based meta-heuristics that emulates the music improvisation process by musicians. In other words, finding a relation between pitches to attain a better state of harmony in music and searching for optimality in optimization process utilizing HS is much alike. This comprehensible nature of HS and its simplicity have led to its application in many optimization problems including the traveling salesperson problem [8], vehicle routing [9], pipe capacity design in water supply networks [10] [11], water network design [12], educational timetabling [13], and shop scheduling [14].

In this paper, we proposed an algorithm combining AFSA with HS to solve the FJSP. AFSA allows an extensive search for the solution space while the HS algorithm is employed to reschedule the results obtained from AFSA based on the new improvised harmony, which will enhance the convergence speed. The objective considered in this paper is to minimize maximal completion time.

The remainder of the paper is organized as follows. An overview of relevant literature is discussed in section 2. The formulation and notation of FJSP are introduced in Section 3. Traditional AFSA and brief standard HS algorithm are described in section 4. In Section 5, the proposed algorithm is presented to solve the FJSP. The computational results and its comparison with other algorithms are shown in section 6. Finally, section 7 provides conclusions and future works.

2. RELATED WORKS

AFSA is one of the preferable methods of optimization among all the swarm intelligence algorithms. This method is inspired by the cooperative movement of the fish and their diverse social behaviors, where it is considered as a parallel and random search optimization algorithm based on simulating fish's behaviors in the water.

Hongwei and Liang [15] proposed an effective artificial fish swarm algorithm with estimation of distribution (AFSA-ED) for obtaining intelligent scheduling strategies. They proposed the pre-principle and post-principle arranging mechanism and an integrated initialization algorithm for enhancing the diversity and modified preying behavior with estimation of distribution and embed attracting behavior to the algorithm for improving the global exploration of the algorithm. Besides, a public factor based critical path search strategy is presented to enhance the local exploitation ability.

Ge, et al. [16] proposed an efficient artificial fish-swarm algorithm with estimation of distribution to solve the flexible job shop scheduling problem with the criterion to minimize the makespan. Considering the interaction of two sub problems, they proposed the pre-principle and post-principle arranging mechanism to adjust the machine assignment and the operation sequence with different orders. For improving the global exploration of the algorithm, they modified preying behavior with estimation of distribution and embed attracting behavior to the algorithm. The critical path based local search was used to balance the exploration and exploitation.

Singh and Mahapatra [17] presented an efficient quantum particle swarm optimization to find near-optimal schedules. The switching operator used in the genetic algorithm is included in QPSO to obviate premature convergence and ameliorate the solution diversity. Furthermore, solution diversity is improved during the use of chaotic numbers (Logistic map) in exchange for random numbers. Utilizing chaotic number in the work provides solution diversity and minifies computational burden.

ASADZADEH [18] presented a parallel and agent-based local search genetic algorithm for solving the job shop scheduling problem. Various agents each with special behaviors is developed to implement the parallel local search genetic algorithm contained in a multi agent system. They parallelize the genetic algorithm using the island model where the population is partitioned into small

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

subpopulations and migration can happen between neighbor subpopulation.

Al-Obaidi and Hussein [19] presented two improved CS algorithms. In the first improvement, the division of the generating solutions instead of the deleted ones speeds up the convergence rate and maintains a good amount of diversification. Because Levy Fight gives the effectiveness of CS algorithm, it proposed in the second improvement, to repeat it multiple times in an acceptable ratio (IR). From the experiments, they found (0.2) of the iterated Levy flight is a suitable ratio for obtaining feasible solutions while increasing this ratio has a time consuming for finding better solutions.

Singh, et al. [20] proposed multiple objective frameworks based on the quantum particle-swarm optimization (QPSO) to create the predictive schedules which can optimize the makespan and the robust measures together at the same time. The results denote that the proposed QPSO algorithm is widely effective in minimizing the makespan in the event which uncertainty is encountered in the terms of stochastic machine breakdown. An exhaustive empirical study is lead up to study the influence of various proposed robustness measures on the produced schedules using benchmark problems.

Alobaidi and Hussein [21] presented an improvement to AFSA algorithm based on VND local search for solving the Flexible Job Shop Scheduling Problem (FJSSP). The Variable Neighborhood Descent (VND) strategy is performed on AFSA by different neighborhood structures to improve and upgrade the performance of the original AFSA. The VND local search used in AFSA-VND provides the algorithms more intensification and exploitation than original AFSA.

In Teekeng, et al. [22], EPSO was proposed to solve flexible job shop scheduling problem (FJSP) based on particle-swarm optimization (PSO). EPSO comprises two groups of features for broadening the solution space of FJSP and obviate precocious convergence to a local optimum. These two groups are as follows: (1) particle life cycle which consists of four features: (i) courting call—increasing number of more efficient offspring (the new solutions), (ii) egg-laying stimulation— increasing number of offspring from best parents (current solutions), (iii) bi-parental reproduction—increasing diversity of the pursue generation (iteration) of solutions, and (iv) population turnover—succeeding the population (current group of all solutions) in the previous generation by a population in a new generation which is as able but is much diverse than the previous one; and (2) discrete position update mechanism—moving particles (solutions) at the flight leader (best solution), namely, interchanging several integers in each solution with these ones in both the best solution and itself, employing similar swarming strategy as the update procedure of the continuous PSO.

Muthiah, et al. [23] proposed a hybridization methodology of the Particle-Swarm Optimization (PSO) and Artificial Bee Colony (ABC). The optimization techniques reduce the time of the makespan of the shops. In the ABC technicality, the scout bee operation depending on the PSO technicality updates the process velocity and position of particles. The Hybrid Algorithm (HA) is elegantly employed to significantly scale down the makespan of the job scheduling function to the least possible. On a close examination and contrast of the outcomes with those of ABC, it is unequivocally established that ABC algorithm is competent to achieve amazing outcomes. The fascinating results substantiate the fact the ground-breaking Hybrid algorithm (HA), Particle swarm optimization (PSO) and Artificial Bee Colony Optimization (ABC) have come out with flying colors by ushering the least make-span interval for all the standard issues addressed.

Finally, and as it is adduced in all the formerly cited studies, different swarm intelligent and meta-heuristics methods are applied to solve the FJSP problem. The challenge is always to have the suitable approach capable for better solving this problem. In the present work, we propose artificial fish swarm algorithm (AFSA) combined with harmony search approach to find better performance than other existing swarm intelligence and meta-heuristics in terms of solution quality.

3. FLEXIBLE JOB SHOP SCHEDULING PROBLEM

The flexible job-shop scheduling problem (FJSP) is a popularization and extension of the traditional job-shop scheduling problem (JSP) in which — prior to the sequencing of operations — an assignment of operations to machines is necessary. A central proposition in classic job shop scheduling (JSP) is that each operation ought to be processed on one predetermined machine. In the fact, the actual relevance of recently flexible job shop scheduling (FJSP) approaches is lying that in practice, where there are more than one machine that is able to process a particular manufacturing task. FJSP was introduced by Brandimarte [24]. Accordingly expanded and generalized the traditional JSP such that for each operation there would be more than one possible machine assignment.

Flexible Job-shop Scheduling Problem (FJSP) can be described as follows. There is a set of n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ to be scheduled.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Each job J_i consists of a predetermined sequence of operations. $O_{i,j}$ is the operation j of job J_i. Each job is operated by a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each machine can process just one operation at a time. Each operation can be processed without interruption during its performance on one of the set of machines. There are no precedence constraints among operations of different jobs. We denote with $P_{i,j,k}$ the processing time of operation $O_{i,j}$ when executed on machine M_k . All machines are available at time 0.

4. ARTIFICIAL FISH SWARM ALGORITHM AND HARMONY SEARCH

In this section, a brief description of AFSA and HS algorithm are given.

4.1. Artificial Fish-Swarm Algorithm (AFSA)

Artificial fish swarm algorithm (AFSA) is an intelligent optimization method based on the metaphor of behavior of the fish swarm. The basic idea of AFSA is elaborated as follows. Since fishes can always find the position full of nutrition by themselves or by following the other fishes, thus the space with the most survival is usually the place that offers the most nutrients. Considering this characteristic, artificial fish algorithm optimizes systems by simulating all kinds of fish actions and combines them with animals' body model [25]. Every artificial fish (AF) in an AFSA framework sets its behavior according to its actual state and its environmental state, so it makes the use of the best position performed by itself and its neighbors. The optimization of AFSA is performed by three behaviors, i.e., preying behavior, swarming behavior, and following behavior.

Suppose $X_i = (X_1, X_2, \ldots, X_n)$ is the current position of AF_i; $Y_i = f(X_i)$ is the fitness function at position X_i which can represent the objective function. *Visual* is the visible distance of AF; *try_number* is the try times of preying behavior; *Step* is the maximum moving step of AF; δ is the crowd factor; nf is the number of AFs within its visual. For AF_i , one target position X_t in its visual can be described by Eq.(1), *Rand*() is a function that generate random numbers in the interval [0,1]. Then the AF_i updates its state by using Eq. (2) when the updating condition is satisfied. The AFSA are summarized by the pseudo code shown in Figure 1 [21].

$$X_t = X_i + visual \cdot Rand \quad (1)$$

$$X_i = X_i + \frac{X_t - X_i}{\|X_t - X_i\|} \cdot Step \cdot Rand() \quad (2)$$

Artificial Fish Swarm Algorithm

Input: *Initialization of population for the problem, visual, try number, crowd factor. Initialization of X_i for each artificial fish AF_i (i =1, 2, …, n)*
Output: *Best Solution*
Evaluate each AF_i, $F(X_i)$ (i =1, 2, …, n)
bulletin = min $F(X_i)$
while (*t <Max Generation*)
  for *each AF_i* **do**
    Perform Follow Behavior on $X_i(t)$ and compute $X_{i,follow}$
    Perform Swarm Behavior on $X_i(t)$ and compute $X_{i,swarm}$
    if *$F(X_{i,follow}) < F(X_{i,swarm})$*
      $X_i(t+1) = X_{i,follow}$
    else
      $X_i(t+1) = X_{i,swarm}$
    end if
  end for
  if *$F(X_{best\_AF}) < F(bulletin)$*
    bulletin= X_{best_AF}
  end if
end while

Figure 1: The AFSA Pseudo Code

The three behaviors of AF are described as follows:

a) Preying: X_t random position chosen within X_i's visible region using Eq.(1). If $Y_t < Y_i$, it moves a step to X_t according to Eq.(2). Otherwise, it chooses another Xt position and determines whether it satisfies the requirement $Y_t < Y_i$. If it is still not satisfied after try_number times, AF_i chooses a random position in its visual region and moves a step towards this direction.

b) Swarming: Let X_c is the center position in the visible region. If the center has more food and low crowd degree as indicated by $Y_c \cdot nf < Y_i \cdot \delta$, then AF_i moves a step towards X_c. Otherwise, AF_i executes default preying behavior.

c) Following: Suppose X_b is the best-found position with high food consistence and low crowd degree. X_b position is in the visible region of X_i position if the position X_b has high food consistence and low crowd degree as indicated by $Y_b \cdot nf < Y_i \cdot \delta$, then AF_i moves a step towards X_b. Otherwise, AF_i executes default preying behavior.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2. Harmony Search Algorithm

HS is a population-based meta-heuristic algorithm, developed in an analogy with musical improvisation. In music performance, each music player improvises one note at a time. All these musical notes are combined together to form a harmony, evaluated by aesthetic standards and improved through practice after practice. In optimization, every variable is assigned a value at a time. All these values are combined together to form a solution vector, evaluated by the objective function and improved iteration by iteration. Due to the rhythm and pitch of each instrument cannot be instantly tuned, perfect harmony is not achieved at first. However, continual practice to enhance the harmony enable the musicians to memorize the specific rhythm and pitch of each instrument, which lead to "good harmony". These collections of "good harmony" are memorized and the unacceptable collections are ignored as superior collections are found. The updating process of the harmony collections continues till the best harmony is obtained. HS performs the process of harmony enhancement and the "good harmony" collections are saved to a solution space called harmony memory (HM), which is HS unique feature compared to other evolutionary optimization algorithms.

HS algorithm contains a solution storage function called HM that necessitates the definition of two parameters: HM considering rate (HMCR) and pitch-adjusting rate (PAR). For more details on HS, please refer to [8] [26]. The major steps in the structure of HS are as follows:
S1: Initialize the algorithm parameters.
S2: Initialize the harmony memory (HM).
S3: Improvise a new solution from the HM.
S4: Update the HM.
S5: Repeat S3 and S4 till the stopping criterion is satisfied.

5. PROPOSED AFSA-HS FOR FJSP

A. Representation

In AFSA-HS, each *AF* represents a feasible solution to the problem. Each AF is expressed by two vectors: machine assignment vector and operation sequence vector, which corresponds to the two sub problems of the FJSP. The machine assignment vector is represented by a vector of N integer values which are the total number of operations. Each element of the vector denotes the machine selection of each operation and a value is which the index of the array of alternative machine set. The operation sequence vector is an un-partitioned permutation with n_i repetitions of job J_i (i = 1,2, ···, n). The length of operation sequence vector equals to N. The index i of job J_i occurs n_i times in the vector, and the k-th occurrence of a job number refers to the k-th operation in the technological sequence of this job. For the problem in Figure 2, a representation:

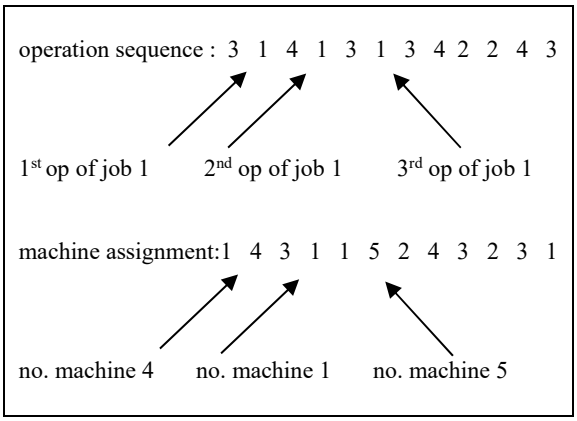

Figure 2: AF Representation Scheme

B. An Improved AFSA Based on Harmony Search algorithm

In this paper, we have proposed an improvement of the original AFSA based on Harmony Memory aiming to enhance the performance of AFSA. Also, we use the HS algorithm with new harmony improvising technique as a strategy for global search in the proposed algorithm without requiring to the HS parameters (PAR and HMCR). Hence we test HS algorithm with AFSA algorithm for solving flexible job shop scheduling problem. This improvising of new harmony is considered as the key vector for providing a good exploration to the entire search space. In the AFSA-HS, the conception of *Visual* represent the maximum number of solution's elements that exchange its location in the solution vectors. Hamming distance is used to compute the distance between the two solutions (X_i and X_j) which is the number of solution's elements that have a different value at corresponding positions. In the discrete space the forward step of AF will be taking the same solution (position) that is evaluated as a next better solution for each AF. This process represents the AF movement. The pseudo code of the proposed algorithm (AFSA-HS) is shown in Figure 3.

The HS in AFSA-HS is applied on the updated AFs obtained in each iteration which are saved in HM. Then they are taken with the global best solution as inputs for more intensification around the best

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

solution area. The pseudo code of Harmony search is shown in Figure 4.

AFSA-HS Algorithm

Input: *Initialization of population for the problem, visual, try number, crowd factor. Initialization of X_i for each artificial fish AF_i (i =1, 2, …, n)*
Output: *Best Solution*
Evaluate each AF_i, $F(X_i)$ (i =1, 2, …, n)
bulletin = min $F(X_i)$
while (*t <Max_Generation*)
  for *each AF_i* **do**
    Perform Follow Behavior on $X_i(t)$ and compute $X_{i,follow}$
    Perform Swarm Behavior on $X_i(t)$ and compute $X_{i,swarm}$
    if *$F(X_{i,follow}) < F(X_{i,swarm})$*
      $X_i(t+1) = X_{i,follow}$
    else
      $X_i(t+1) = X_{i,swarm}$
    end if
  end for
  Find G_{best_AF} which has the best $F(X_i^{(t+1)})$ (i =1, 2, …, n)
  HM= updated AFs obtained in t^{th} generation
  Perform HS Algorithm on G_{best_AF} and compute G_{best_HM}
  if *$F(G_{best\_HM}) < F(G_{best\_AF})$*
    $G_{best\_AF} = G_{best\_HM}$
  end if
  if *$F(G_{best\_AF}) < F(bulletin)$*
    bulletin= G_{best_AF}
  end if
end while

Figure 3: Pseudo Code Of The Proposed AFSA-HS Algorithm For FJSP

In HS, the available members in the memory are used to construct new harmony at each generation. Due to this reason, there is no need for an extra memory where the population members corresponding to the earlier generations are stored. A new solution vector is generated which is computed using the AFs in the memory by making some swapping among the solution elements of the *i*th AF randomly. The distance between new harmony and *i*th AF is less or equal to *Visual*. Then the global best solution is compared to the new improvised vector. If the new vector is better than global best solution, the algorithm will progress to the next step of replacing the *i*th solution stored in the HM with the new vector. Otherwise the *i*th solution is kept in HM. Hence another new vector is generated from another AF_{i+1} (*i*+1th member) stored in the HM and so on for all AFs stored in HM. After the search process is completed by all employed AFs in HM, we compare the global best solution to the global best solution stored in HM. Then the better one is recorded in bulletin board. Since this process has different improvising vectors, it maintains some diversity within the search and gives the proposed AFSA-HS algorithm more exploitation around global best solution. Also an enhancing in speed of the convergence rate is achieved.

Harmony Search Algorithm

Input: *Current best AF Solution G_{best_AF}.*
Set AF index i = 1
Output: *G_{best_HM}* /* Global best solution in Harmony Memory */
while *stopping criterion is not satisfied /* each AF in HM */*
  / Generate a new solution */*
    Pick the ith value from the solutions in HM
    / New solution generated */*
    Pertub the value picked
  if *new solution better than the G_{Best_AFSA} solution (in terms of fitness)*
    Replace the ith solution in HM with new solution
  end if
  Increment the AF index i = i + 1
end while
Find G_{best_HM} which has the best $F(X_i^{(t+1)})$ in HM

Figure 4: The HS Algorithm Pseudo Code

6. EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the performance of the AFSA-HS for FJSP, we consider three sets of well-known benchmark with 30 instances from Hurink, et al. [27] [28]. Hurink data sets are three: edata, rdata, and vdata. The jobs number ranges in each set from 6 to 30, machines number ranges between 4 and 15. While the flexibility of each operation in the three data sets edata, rdata, and vdata are 1.15, 2 and 2-7.5 respectively. In the original AFSA algorithm, the moving step of the Artificial Fish is done in continuous state space. Therefore, we adapt the AFSA and AFSA-HS algorithms to be suitable for searching in discrete state space of the problem by some neighborhood strategies.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The AFSA and AFSA-HS are coded and implemented in MATLAB R2013a language on an Intel Core i5 2.3 GHz. personal computer with 4GB of RAM. The AFSA and AFSA-HS are applied to the same instances of FJSP with the same value of parameters. The algorithms AFSA and AFSA-HS run 5 independent times for each instance of edata, rdata, and vdata. In order to obtain meaningful results we calculated the average of the 5 test runs. The measured performance is close to the lower bound of the benchmark. To illustrate the quality of the results obtained by the AFSA and AFSA-HS, the relative error (RE) is introduced. RE is calculated by the following Eq. (3):

$$RE(\%) = \frac{AMS - LB}{LB} \times 100 \quad (3)$$

where AMS is the Average makespan for each instance obtained by the corresponding algorithms. LB is the best-known solution or the lower bound of the benchmark.

In Table 1 comparisons of makespan obtained by AFSA and our proposed AFSA-HS on thirty FJSP instances from Hurink data set are introduced. The 1st column represents the name of the problem. The 2nd column represents the size of the problem. The 3rd column points to the lower bound of the benchmark. The fourth and sixth columns represent the average of makespan resulted from AFSA and AFSA-HS respectively. Table 1 shows the best results from our proposed algorithm for all 30 test instances of the Hurink data sets (edata, rdata, and vdata) compared to the best results obtained from the artificial fish-swarm algorithm (AFSA). Where the compared results are either the same or better than the best solutions from AFSA or equal to the lower bound.

Our algorithm is different in its simplicity and it is far from complexity and cost. It increases the diversity of the solutions by improvising a new harmony from each artificial fish searching for a perfect state of harmony in terms of fitness. Thus it produces a new solution around the existing solution quality. The best harmonies will be carried over to the harmony memory. Therefore we may get more than one new harmony with fitness better than the fitness of the global best solution. This led to an improvement in the quality of the solution and the speed of convergence towards the optimal solution.

Table 1: The Average Of Makespan Results Of Hurink Data Sets By AFSA And AFSA-HS Algorithm

Instances	Size n x m	LB	***AFSA***	***AFSA-HS***
			AMS	*AMS*
edata-mt06	6 x 6	**55**	**55**	**55**
edata-mt10	10 x 10	871	985	**912**
edata-la1	10 x 5	**609**	620	**609**
edata-la2	10 x 5	655	691	**658**
edata-la3	10 x 5	550	578	**564**
edata-la4	10 x 5	568	606	**590**
edata-la5	10 x 5	**503**	518	**503**
edata-la6	15 x 5	**833**	860	**833**
edata-la7	15 x 5	762	811	**778**
edata-la8	15 x 5	845	869	**851**
rdata-mt06	6 x 6	**47**	48	**47**
rdata-mt10	10 x 10	679	848	**754**
rdata-la1	10 x 5	570	605	**582**
rdata-la2	10 x 5	529	563	**544**
rdata-la3	10 x 5	477	509	**488**
rdata-la4	10 x 5	502	540	**516**
rdata-la5	10 x 5	457	482	**466**
rdata-la6	15 x 5	799	827	**804**
rdata-la7	15 x 5	749	787	**755**
rdata-la8	15 x 5	765	797	**770**
vdata-mt06	6 x 6	**47**	**47**	**47**
vdata-mt10	10 x 10	655	773	**677**
vdata-la1	10 x 5	570	599	**577**
vdata-la2	10 x 5	529	579	**538**
vdata-la3	10 x 5	477	508	**486**
vdata-la4	10 x 5	502	536	**511**
vdata-la5	10 x 5	457	484	**469**
vdata-la6	15 x 5	799	829	**807**
vdata-la7	15 x 5	749	781	**755**
vdata-la8	15 x 5	765	785	**770**

The bold style of data mean the best results among the compared algorithms or the ideal makespan

The comparison between the proposed AFSA-HS algorithm and the results obtained by other algorithms is expressed in Table 2. In this table, comparisons of the average percentage of relative error from the lower bound of our proposed AFSA-HS algorithm to the average percentages of relative error of the (CS-BNG, and CS-ILF) [19] and AFSA-VND [21] algorithms on thirty FJSP instances from Hurink data sets are prepared. The fifth, eighth, eleventh and fourteenth columns represent the relative error. In addition, the sixth, ninth, twelfth, and fifteenth columns represent the percentage of improvement. The percentage improvement (PI) for makespan using AFSA-HS over AFSA is defined as follows:

$$PI(\%) = \frac{AMS_{AFSA} - AMS_{AFSA\_HS}}{AMS_{AFSA}} \times 100 \quad (4)$$

Furthermore Table 2 indicates that a maximum of percentage improvement attainable by AFSA-HS is 12.42 with a vdata mt10 whilst it is 4.84 by AFSA-VND with a vdata la2 for the benchmark problems considered in this study.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Comparison Of Percentage Relative Errors And Percentage Improvement Obtained By Using AFSA-HS, AFSA-VND, CS-BNG, And CS-ILF On Hurink Data Sets

Instances	Size n x m	LB	***AFSA-HS***			***AFSA-VND***			***CS-BNG***			***CS-ILF***		
			AMS	*RE(%)*	*PI(%)*	*AMS*	*RE(%)*	*PI(%)*	*AMS*	*RE(%)*	*PI(%)*	*AMS*	*RE(%)*	*PI(%)*
edata-mt06	6 x 6	**55**	**55**	**0.00**	0.00	**55**	**0.00**	0.00	**55**	**0.00**	1.79	**55**	**0.00**	1.79
edata-mt10	10 x 10	871	**912**	0.05	7.41	953	0.09	3.25	986	0.13	17.21	979	0.12	17.80
edata-la1	10 x 5	**609**	**609**	**0.00**	1.77	617	0.01	0.48	636	0.04	12.76	634	0.04	13.03
edata-la2	10 x 5	655	**658**	0.00	4.78	686	0.05	0.72	707	0.08	9.71	694	0.06	11.37
edata-la3	10 x 5	550	**564**	0.03	2.42	568	0.03	1.73	593	0.08	11.09	588	0.07	11.84
edata-la4	10 x 5	568	**590**	0.04	2.64	598	0.05	1.32	620	0.09	12.55	619	0.09	12.69
edata-la5	10 x 5	**503**	**503**	**0.00**	2.90	511	0.02	1.35	525	0.04	13.22	526	0.05	13.06
edata-la6	15 x 5	**833**	**833**	**0.00**	3.14	850	0.02	1.16	864	0.04	11.48	861	0.03	11.78
edata-la7	15 x 5	762	**778**	0.02	4.07	799	0.05	1.48	818	0.07	14.79	819	0.07	14.69
edata-la8	15 x 5	845	**851**	0.01	2.07	860	0.02	1.04	880	0.04	12.09	868	0.03	13.29
rdata-mt06	6 x 6	**47**	**47**	**0.00**	2.08	**47**	**0.00**	2.08	55	0.17	0.00	55	0.17	0.00
rdata-mt10	10 x 10	679	**754**	0.11	11.08	824	0.21	2.83	802	0.18	24.84	801	0.18	24.93
rdata-la1	10 x 5	570	**582**	0.02	3.80	590	0.04	2.48	607	0.06	16.04	609	0.07	15.77
rdata-la2	10 x 5	529	**544**	0.03	3.37	554	0.05	1.60	573	0.08	15.74	567	0.07	16.62
rdata-la3	10 x 5	477	**488**	0.02	4.13	497	0.04	2.36	518	0.09	16.59	512	0.07	17.55
rdata-la4	10 x 5	502	**516**	0.03	4.44	528	0.05	2.22	542	0.08	16.10	538	0.07	16.72
rdata-la5	10 x 5	457	**466**	0.02	3.32	476	0.04	1.24	484	0.06	16.12	480	0.05	16.81
rdata-la6	15 x 5	799	**804**	0.01	2.78	816	0.02	1.33	832	0.04	14.58	821	0.03	15.71
rdata-la7	15 x 5	749	**755**	0.01	4.07	767	0.02	2.54	779	0.04	15.05	776	0.04	15.38
rdata-la8	15 x 5	765	**770**	0.01	3.39	788	0.03	1.13	793	0.04	15.46	790	0.03	15.78
vdata-mt06	6 x 6	**47**	**47**	**0.00**	0.00	**47**	**0.00**	0.00	49	0.04	10.91	48	0.02	12.73
vdata-mt10	10 x 10	655	**677**	0.03	12.42	814	0.14	-5.30	746	0.14	25.40	729	0.11	27.10
vdata-la1	10 x 5	570	**577**	0.01	3.67	587	0.04	2.00	613	0.08	15.80	609	0.07	16.35
vdata-la2	10 x 5	529	**538**	0.02	7.08	551	0.06	4.84	565	0.07	16.30	564	0.07	16.44
vdata-la3	10 x 5	477	**486**	0.02	4.33	497	0.05	2.17	515	0.08	17.86	520	0.09	17.07
vdata-la4	10 x 5	502	**511**	0.02	4.66	528	0.05	1.49	534	0.06	18.10	531	0.06	18.56
vdata-la5	10 x 5	457	**469**	0.03	3.10	472	0.05	2.48	485	0.06	17.38	499	0.09	14.99
vdata-la6	15 x 5	799	**807**	0.01	2.65	814	0.03	1.81	826	0.03	15.80	821	0.03	16.31
vdata-la7	15 x 5	749	**755**	0.01	3.33	763	0.03	2.30	774	0.03	17.75	773	0.03	17.85
vdata-la8	15 x 5	765	**770**	0.01	1.91	786	0.02	-0.13	779	0.02	18.17	787	0.03	17.33
ARE(%)				**0.02**			**0.04**			**0.07**			**0.06**	

The bold style of data mean the best results among the compared algorithms

In addition, this table also indicates the percentage improvement of CS which attainable by CS-BNG is 25.4 with a vdata mt10 whereas it is 27.1 by CS-ILF with a vdata mt10 for the same benchmark problems. Furthermore, the average percentage of relative error (ARE) of the four algorithms (AFSA-HS, AFSA-VND, CS-BNG, and CS-ILF) for the 30 test instances are 0.02, 0.04, 0.07, and 0.06, respectively. Although the percentage improvement of CS-ILF for the original CS algorithm was the best, AFSA-HS obtained a better solution quality comparing with the other improvements. According to the best makespans from Table 2, it can be seen that the best results obtained by AFSA-HS are equal or better than that of other algorithms when dealing with almost all of the 30 Hurink data instances. Our AFSA-HS outperforms AFSA-VND in 27 out of the 30 Hurink data instances. Also it outperforms CS-BNG and CS-ILF in 29 out of the 30 Hurink data instances. Table 3 produces the iterations number of the proposed AFSA-HS and other compared algorithms. To such instances, from tables 1 and 3, it can be seen that our AFSA-HS obtains the ideal makespans (55, 609, 503, 833, 47, and 47) with 2, 101, 223, 45, 242, and 12 iterations respectively. The basic AFSA obtains the ideal makespans (55 and 47) with 8 and 63 iterations. Though in tables 2 and 3, the AFSA-VND can obtain the ideal makespans (55, 47, and 47), however, it needs as many as 6, 403, and 30 iterations respectively. The CS-BNG and CS-ILF only obtain the ideal makespan (55) and they need 39 and 69 iterations. Therefore, it is concluded that our AFSA-HS has more powerful optimizing ability in dealing with the flexible job shop scheduling problem. In terms of the speed, 80% of the results obtained from the benchmark instances by the AFSA-VND are faster than that of the original AFSA. Whereas 60% of the results obtained from the benchmark instances by the AFSA-HS are faster than the original AFSA. Although AFSA-HS outperform the AFSA-VND in terms of the solution quality, approximately 20% of the results obtained from the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

benchmark instances by AFSA-HS are slower compared with AFSA-VND. This result is attributed to slightly increasing in computational time.

Table 3: Comparison Of Iterations Number Of AFSA, AFSA-HS, AFSA-VND, CS, CS-BNG, And CS-ILF On Hurink Data Sets

Instances	Size n x m	AFSA iterations no.	AFSA-HS iterations no.	AFSA-VND iterations no.	CS iterations no.	CS-BNG iterations no.	CS-ILF iterations no.
edata-mt06	6 x 6	8	2	6	444	39	69
edata-mt10	10 x 10	409	416	528	452	718	466
edata-la1	10 x 5	445	101	238	475	352	369
edata-la2	10 x 5	472	187	303	540	316	293
edata-la3	10 x 5	442	577	568	507	399	477
edata-la4	10 x 5	644	48	627	584	432	433
edata-la5	10 x 5	450	223	403	653	370	199
edata-la6	15 x 5	569	45	217	498	519	460
edata-la7	15 x 5	667	191	528	546	674	490
edata-la8	15 x 5	556	433	217	488	576	550
rdata-mt06	6 x 6	351	242	403	482	136	406
rdata-mt10	10 x 10	548	825	515	569	726	792
rdata-la1	10 x 5	620	696	368	676	532	437
rdata-la2	10 x 5	694	618	571	528	498	638
rdata-la3	10 x 5	509	468	497	503	527	615
rdata-la4	10 x 5	595	160	574	397	538	596
rdata-la5	10 x 5	862	175	386	387	577	651
rdata-la6	15 x 5	489	177	379	591	393	601
rdata-la7	15 x 5	652	728	632	458	440	692
rdata-la8	15 x 5	566	723	513	595	402	482
vdata-mt06	6 x 6	63	12	30	449	86	132
vdata-mt10	10 x 10	760	809	735	648	713	678
vdata-la1	10 x 5	637	358	551	425	497	643
vdata-la2	10 x 5	53	198	61	510	656	581
vdata-la3	10 x 5	536	902	777	468	586	450
vdata-la4	10 x 5	610	680	377	412	529	577
vdata-la5	10 x 5	595	81	454	540	592	363
vdata-la6	15 x 5	633	472	859	425	769	650
vdata-la7	15 x 5	451	583	349	692	606	681
vdata-la8	15 x 5	640	914	244	479	650	520

The essential difference of the AFSA-HS, AFSA-VND, CS-BNG and CS-ILF is in the selection procedure. This difference in selection method would be reflected in the quality of solutions. AFSA-HS is different from that of AFSA-VND, CS-BNG, and CS-ILF in that it needs fewer parameters and can be executed easily. In order to determine the statistical differences between the AFSA-HS and the compared algorithms, the Friedman test is conducted. The results are presented in Table 4. It can be seen from the Friedman test results that the differences among the four algorithms are statistically relevant with 97% certainty. The AFSA-HS obtains the best overall rank. The proposed algorithm has outperformed almost all the benchmark instances. Therefore, it is concluded from the computational results that the proposed AFSA-HS provides better performance than those testified by other algorithms.

Table 4: Friedman Test Of Different Algorithms

Algorithm	Rank	1-*p* value	χ^2	Diff.?
AFSA-HS	1.08	0.97	69.31	Yes
AFSA-VND	2.15			
CS-BNG	3.67			
CS-ILF	3.10			

7. CONCLUSIONS AND FUTURE WORKS

In this paper, the proposed AFSA-HS algorithm is constructed to be a good problem-solving technique for scheduling problem with the criterion to minimize the makespan. In order to improve the exploration of the original AFSA algorithm, the harmony search (HS) is exploited in AFSA-HS. It is based on the new improvised harmony from results obtained by AFSA. Hence AFSA-HS provides more intensification than the original AFSA. In this paper,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

we aim to implement an efficient algorithm which can be easily reconfigured for embedded systems capable of making real-time decisions according to the state of resources and any unplanned or unforeseen events. Harmony search has a good diversity. Furthermore its integration with the artificial fish swarm algorithm results in an improved solution diversity of artificial fish swarm algorithm. Although solution diversity has increased, the computational time has increased slightly too. In addition, the proposed algorithm includes the parameters of both artificial fish swarm algorithm and harmony search algorithm resulting in a little increment in computational time. Here 30 benchmark problems are taken into consideration to assess the performance process in the FJSP. The computational results and comparisons prove that the proposed AFSA-HS outperforms several existing algorithms and it is effective for FJSP.

This study can be extended in future to handle more complex FJSP with multi-objective functions. In addition to the maximum completion time (Cmax) that is used to measure the performance of AFSA-HS algorithm. Another multi-objective functions such as the workload of the critical machine, the total workload of all machines, Tardiness time, and Flow time could be used for the same purpose. The proposed algorithm could be applied to other optimization problems. One can consider applying AFSA with its improvement to solve the FJSP in case of the occurrence of different disruptions such as machines breakdown or raw materials shortages or others. The first population plays an important role in solution diversity of artificial fish swarm algorithm. We suggest a method for generating the initial population with a high level of quality to ensure good randomization and high diversity. This leads to reach the optimal solution in less time.

REFERENCES

[1] J. Tang, G. Zhang, B. Lin and B. Zhang, "A hybrid algorithm for flexible job-shop scheduling problem," *Procedia Engineering,* vol. 15, pp. 3678-3683, 2011.

[2] P.-J. Lai and H.-C. Wu, "Using heuristic algorithms to solve the scheduling problems with job-dependent and machine-dependent learning effects," *Journal of Intelligent Manufacturing,* vol. 26, no. 4, pp. 691-701, 2015.

[3] K.-Z. Gao, P. N. Suganthan, Q.-K. Pan, T. J. Chua, T. X. Cai and C.-S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *Journal of Intelligent Manufacturing,* vol. 27, no. 2, pp. 363-374, 2016.

[4] M. Gen and L. Lin, "Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey," *Journal of Intelligent Manufacturing,* vol. 25, no. 5, pp. 849-866, 2014.

[5] A. Miguel, F. P{\'e}rez, M. Fernanda and P. Raupp, "A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing,* vol. 27, no. 2, p. 409, 2016.

[6] R. Azizi, "Empirical study of artificial fish swarm algorithm," *arXiv preprint arXiv:1405.4138,* 2014.

[7] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M. N. Bilbao, S. Salcedo-Sanz and Z. W. Geem, "A survey on applications of the harmony search algorithm," *Engineering Applications of Artificial Intelligence,* vol. 26, no. 8, pp. 1818-1831, 2013.

[8] Z. W. Geem, J. H. Kim and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *simulation,* vol. 76, no. 2, pp. 60-68, 2001.

[9] Z. W. Geem, K. S. Lee and Y. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences,* vol. 2, no. 12, pp. 1552-1557, 2005.

[10] Z. W. Geem and C.-L. Tseng, "Engineering Applications of Harmony Search.," in *GECCO Late Breaking Papers*, 2002.

[11] Z. W. Geem, "New methodology, harmony search and its robustness," *Late-Breaking Paper of Genetic and Evolutionary Computation Confrence 2002, New York, NY,* 2002.

[12] Z. W. Geem, "Particle-swarm harmony search for water network design," *Engineering Optimization,* vol. 41, no. 4, pp. 297-311, 2009.

[13] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research,* vol. 194, no. 1, pp. 3-31, 2012.

[14] L. Wang, Q.-K. Pan and M. F. Tasgetiren, "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem," *Computers \& Industrial Engineering,* vol. 61, no. 1, pp. 76-83, 2011.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[15] G. Hongwei and S. Liang, "Intelligent scheduling in flexible job shop environments based on artificial fish swarm algorithm with estimation of distribution," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, 2016.

[16] H. Ge, L. Sun, X. Chen and Y. Liang, "An Efficient Artificial Fish Swarm Model with Estimation of Distribution for Flexible Job Shop Scheduling," *International Journal of Computational Intelligence Systems,* vol. 9, no. 5, pp. 917-931, 2016.

[17] M. R. Singh and S. S. Mahapatra, "A quantum behaved particle swarm optimization for flexible job shop scheduling," *Computers \& Industrial Engineering,* vol. 93, pp. 36-44, 2016.

[18] L. ASADZADEH, "SOLVING THE JOB SHOP SCHEDULING PROBLEM WITH A PARALLEL AND AGENT-BASED LOCAL SEARCH GENETIC ALGORITHM.," *Journal of Theoretical \& Applied Information Technology,* vol. 62, no. 2, 2014.

[19] A. T. S. Al-Obaidi and S. A. Hussein, "Two Improved Cuckoo Search Algorithm to Solve Flexible Job-Shop Scheduling Problem," *International Journal on Perceptive and Cognitive Computing,* vol. 2, no. 2, 2016.

[20] M. R. Singh, S. Mahapatra and R. Mishra, "Robust scheduling for flexible job shop problems with random machine breakdowns using a quantum behaved particle swarm optimisation," *International Journal of Services and Operations Management,* vol. 20, no. 1, pp. 1-20, 2014.

[21] A. T. S. Alobaidi and S. A. Hussein, "An improved Artificial Fish Swarm Algorithm to solve flexible job shop," in *New Trends in Information \& Communications Technology Applications (NTICT), 2017 Annual Conference on*, 2017.

[22] W. Teekeng, A. Thammano, P. Unkaw and J. Kiatwuthiamorn, "A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization," *Artificial Life and Robotics,* vol. 21, no. 1, pp. 18-23, 2016.

[23] A. Muthiah, A. Rajkumar and R. Rajkumar, "Hybridization of Artificial Bee Colony algorithm with Particle Swarm Optimization algorithm for flexible Job Shop Scheduling," in *Energy Efficient Technologies for Sustainability (ICEETS), 2016 International Conference on*, 2016.

[24] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations research,* vol. 41, no. 3, pp. 157-183, 1993.

[25] Z. Li, H. Zhang, J. Xu and Q. Zhai, "Recognition and localization of harmful acoustic signals in wireless sensor network based on artificial fish swarm algorithm," *Journal of Theoretical and Applied Information Technology,* vol. 49, no. 1, 2013.

[26] J. H. Kim, Z. W. Geem and E. S. Kim, "Parameter estimation of the nonlinear Muskingum model using harmony search," *JAWRA Journal of the American Water Resources Association,* vol. 37, no. 5, pp. 1131-1138, 2001.

[27] D. Behnke and M. J. Geiger, "Test instances for the flexible job shop scheduling problem with work centers," 2012.

[28] J. Hurink, B. Jurisch and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *Operations-Research-Spektrum,* vol. 15, no. 4, pp. 205-215, 1994.