Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

An improved sheep flock heredity algorithm for job shop scheduling and flow shop scheduling problems

Chandramouli Anandaraman

Department of Production Engineering, National Institute of Technology, Tiruchirappalli – 620 015, Tamil Nadu, India

A R T I C L E I N F O

Article history:
Received 10 March 2010
Received in revised form
June, 23, 2011
Accepted 24 June 2011
Available online
24 June 2011

Keywords:
Job shop scheduling
Flow shop scheduling
Sheep flock heredity algorithm
Makespan

A B S T R A C T

Job Shop Scheduling Problem (JSSP) and Flow Shop Scheduling Problem (FSSP) are strong NP-complete combinatorial optimization problems among class of typical production scheduling problems. An improved Sheep Flock Heredity Algorithm (ISFHA) is proposed in this paper to find a schedule of operations that can minimize makespan. In ISFHA, the pairwise mutation operation is replaced by a single point mutation process with a probabilistic property which guarantees the feasibility of the solutions in the local search domain. A Robust-Replace (R-R) heuristic is introduced in place of chromosomal crossover to enhance the global search and to improve the convergence. The R-R heuristic is found to enhance the exploring potential of the algorithm and enrich the diversity of neighborhoods. Experimental results reveal the effectiveness of the proposed algorithm, whose optimization performance is markedly superior to that of genetic algorithms and is comparable to the best results reported in the literature.

1. Introduction

Scheduling theory deals with formulation and study of various scheduling models. It also includes development of associated solution techniques (Wiers, 1997). Some widely studied classical models comprise single machine, parallel machine, flowshop scheduling and job shop scheduling models (Baker, 1974; Chandrasekaran et al., 2005; Fink & Vob, 2003). The objective of JSSP and FSSP is to find a permutation schedule that minimizes the maximum completion time of a sequence. The FSSP constitutes k jobs in an m-machine flow-shop, where every job has m operations and every machine has k jobs. Every job executes its operations on every machine in the order of (M_1, M_2, ..., M_m). For $n \times m$ JSSP, the fundamental set of possible schedules is $(n!)^m$. The difficulty in the scheduling problems lies in finding the best possible schedule from a number of schedules. The computational time for algorithms searching possible solution space to identify a best possible schedule increases exponentially with problem size. Hence, JSSP and FSSP belong to the set of problems classified as NP-complete problems (Baker, 1974; French, 1982; Giffler & Thompson, 1960; Garey et al., 1976).

During the last decade, many evolutionary algorithms have been widely applied in the area of production scheduling, especially JSSP and FSSP. A computational method for JSSP was proposed,

* Corresponding author. Tel: +919840557585
E-mail: chandramouli.anandaraman@gmail.com (C. Anandaraman)

© 2011 Growing Science Ltd. All rights reserved.
doi: 10.5267/j.ijiec.2011.06.007

anchored in the principles of artificial immune system (Chandrasekaran et al., 2005). The objective considered was to find the optimal makespan values. A hybrid simulated annealing algorithm based on a novel immune mechanism was proposed for the JSSP (Zhang & Wu, 2010). Here, the immune procedure was combined with a simulated annealing algorithm with the objective of minimizing total weighted tardiness. A new hybrid swarm intelligence algorithm was presented (Lin et al., 2010). The algorithm consists of particle swarm optimization (PSO), simulated annealing (SA) technique and multi-type individual enhancement scheme. A genetic algorithm based approach was presented in which a hormone modulation mechanism was applied (Wang & Tang, 2011). In this approach, the hormone modulation was applied to improve the performance of the algorithm. An efficient evolutionary approach based on the classical SFHA (CSFHA) was applied to compute optimal schedules for the JSSP (Vikram & Chandramouli, 2011).

A new hybrid PSO model named HPSO that combines random-key (RK) encoding scheme, individual enhancement (IE) scheme, and PSO was presented and used to solve the FSSP (Kuo et al., 2009). A hybrid alternate two phases PSO algorithm called ATPPSO was proposed to solve the FSSP with the objective of minimizing makespan, which combines the PSO with genetic operators and annealing strategy (Zhang et al., 2010). A discrete firefly meta-heuristic with local search was built for makespan minimization in permutation FSSP (Sayadi et al., 2010). A hybrid modified global-best harmony search (hmgHS) algorithm for solving the blocking permutation flow shop scheduling problem with the makespan criterion was proposed with objective of minimizing makespan (Wang et al., 2011). The problem of scheduling jobs in a no-wait flowshop problem with sequence-dependent setup times with the objective of minimizing makespan was addressed (Araújo & Nagano, 2011), where a new constructive heuristic named Gap-heuristic (GAPH) based on a structural property was proposed.

Generally, an evolutionary algorithm exhibits parallelism, and contains certain redundancy of past solutions. It is not easy to regulate an evolutionary algorithm's convergence, so it often suffers from premature convergence (Leung et al., 1997; Yang & Donglas, 1998; Rudolph, 1994), and it is also difficult to choose suitable parameters for it (Grefenstette, 1988). It is still important to enhance the performance of such evolutionary algorithms. In this paper, an improved SFHA is proposed for the JSSP and FSSP to enhance the exploring potential of the evolutionary algorithm and to enrich the diversity of the search domain. The improvement in both the local search procedure and the global search procedure helps in increasing the overall efficiency of the algorithm without any compromise on the computational time and convergence. The local search procedure is enhanced by introducing a single point mutation process in place of pairwise mutation process, whereas the global search procedure is augmented by the introduction of a new R-R heuristic. In this way, the scheduling problems are solved efficiently.

2. Sheep flock heredity algorithm

2.1. Introduction

The Sheep Flock Heredity Algorithm was initially designed and was applied to scheduling problems (Nara et al., 1999). The algorithm was based on the natural evolution of sheep in a flock. The chromosomes represent machine schedules.

2.2. Algorithm Description

Normally, sheep in each flock are living within their own flock under the control of shepherds. Therefore, the genetic inheritance only occurs within the flock. Some special characteristics in one flock develop only within the flock by heredity, and the sheep with high fitness characteristics to their environment breed in the flock. Assume there are two flocks occasionally mixed with the other flocks. The characteristics of the sheep in neighboring flocks can be inherent to the sheep in this flock. In the field, the flock of the sheep, which has better characteristics to the field environment, breeds most.

The natural evolution phenomenon of flocks can be corresponded to the genetic operations of this type of string. For this kind of string, we can define two kinds of operations. (i) Normal genetic operations between strings. (ii) Genetic operations between sub-strings within one string. In SFHA, special string structure and hierarchical genetic operations (crossover and mutation) are introduced. They are (i) sub-chromosome level genetic operation and (ii) chromosome (global) level genetic operation. This hierarchical operation is referred to as “multi-stage-genetic operation”.

3. Improved sheep flock heredity algorithm

3.1. Introduction

Based on the mechanics of natural selection and genetics, evolutionary algorithms such as genetic algorithms combine the concept of survival of the fittest among solutions with a structured yet randomized information exchange and offspring creation. These algorithms are naturally parallel and exhibit implicit parallelism, which do not evaluate and improve a single solution but analyze and modify a set of solutions simultaneously (Goldberg, 1989). The possibility of the solution being trapped in a local optimum is considerably reduced by the ability of the algorithm to operate on many solutions simultaneously and gather information from all current solutions to direct the search.

Moreover, these evolutionary algorithms can retain useful redundant information about what is learned from previous searches by their representation of individuals in the population. Critical components of past good solutions can be captured, which can be combined via crossover to form high quality solutions. Unfortunately, they may lose solutions and substructures because of disruptive effects of genetic operators, and it is not easy to regulate the algorithm’s convergence so that a pure SFHA may easily produce premature and poor results. To enhance the performance of genetic searches and improve convergence, an improved SFHA (ISFHA) is proposed. In contrast to the classical SFHA (CSFHA), the ISFHA has superior features. First, the pairwise mutation process is replaced by a single mutation process. In a CSFHA, the pairwise mutation process performs a limited local search and introduces some diversity, but it is hard to control its behavior whereas in an ISFHA, the mutation rate of the single point mutation is much easier to control thereby enhancing the local search behavior.

A new robust-replace (R-R) heuristic is introduced to enrich the neighbor search templates so that a more powerful exploring ability and a larger exploring region can be attained. This heuristic helps in improving the solution search of the algorithm in the global domain. Thus, by the introduction of these two processes into the algorithm, the limited local search of mutation is enhanced as well as a constant search of solution between local and global search domains is established. Such a procedure can also be executed in a parallel mode to reduce serial search time. In addition, since the ISFHA retains the generality of SFHA, it can be implemented easily and applied to any combinatorial or functional optimization problems. If necessary, ISFHA can be converted easily to CSFHA or GA by adjusting the framework.

3.2. Processes in ISFHA

The ISFHA consists of the following processes.

1. Sub chromosomal level crossover

2. Sub chromosomal level mutation

    a. Inverse mutation

   b. Single point mutation

3. Robust-replace heuristic

4. Chromosomal level mutation

   a. Inverse mutation

   b. Single point mutation

3.2.1. Sub chromosomal level crossover

In sub chromosomal level crossover, the entire chromosome is split into sub chromosomes of equal length. These sub chromosomes are moved randomly within the chromosome to new positions, thus forming a new chromosome.

3.2.2. Sub chromosomal level mutation

The mutation consists of two steps, inverse mutation and single point mutation.

(a) Inverse mutation

In a sequence, two random positions are selected. The portion of the sequence between these two positions is inverted to get a new mutated sequence.

(b) Single point mutation

A random position is selected in the sequence and moved to another random position in the sequence. As the mutation occurs with respect to a single point, the process is called single point mutation.

3.2.3. Robust-Replace heuristic

Each schedule (chromosome) has a makespan value that refers to the amount of robustness of that chromosome. The robustness of each schedule is calculated from the robust factor. The robust factor is defined as given in Eq. (1).

$$\text{Robust Factor} = 1 / \text{makespan} \quad (1)$$

From this relation, a lower makespan value gives a higher robust factor value. The average robust factor is calculated for the population. The schedules with robust factor values less than the average robust factor value are replaced by new schedules. This mechanism allows finding new schedules that correspond to new search regions in the total search space.

3.2.4. Chromosomal level mutation

The chromosome level mutation is performed among all the chromosomes in the population through inverse mutation and single point mutation.

3.3. Steps in ISFHA

Begin: Initialize the population.

Stage 1:

Select the parent.

Sub chromosome level crossover:

Set sub chromosome level crossover probability

```
    If population is less than or equal to sub chromosome level probability
        Perform sub chromosome level crossover
    Else retain the old sequences
Sub chromosome level mutation:
Set sub chromosome mutation probability
    If population probability is less than or equal to sub chromosome mutation probability
        Perform sub chromosome level mutation
    Else retain the same sequences.
Stage 2:
Robust-Replace heuristic:
Replace low robust chromosomes with new chromosomes
Stage 3:
Chromosome level mutation:
    If population probability is less than or equal to mutation probability
        Perform chromosome level mutation
    Else retain the same sequences
End if terminal condition satisfied
```

3.4. Flow chart of ISFHA

The flow chart of ISFHA is given in Fig. 1.

4. Problem description

In this work, the problems considered for optimization are job shop scheduling and flow shop scheduling problems. The objective is to minimize makespan, which is the maximum completion time of all operations.

4.1. Job Shop Scheduling Problem

4.1.1. Problem Description

A job shop consists of *n* jobs and *m* machines. Each job must go through *m* machines to complete its sequence of operations. Each operation uses one of *m* machines to complete a job's work. In general, one job being processed on one machine is considered as one operation noted O_{ji} (means *j*th job being processed on *i*th machine, $1 \leq j \leq n$, $1 \leq i \leq m$). The objective of JSSP is to find an appropriate schedule of all machines that can minimize the makespan.

4.1.2. Assumptions

In a JSSP instance, a set of *n* jobs are waiting to be processed on a set of *m* machines under the following basic assumptions:

- Machine breakdown does not occur, which means all the machines are continuously available throughout the production stage.

- Pre-emption of operations is not allowed, which means the processing of an operation cannot be interrupted once started.
- The transportation time to deliver relevant jobs between different machines is neglected.
- The setup time for the machines to switch between different jobs is neglected.
- Each machine can process only one job at a time.
- Each job can be processed by only one machine at a time.
- The sequence of operations of every job should be predefined.

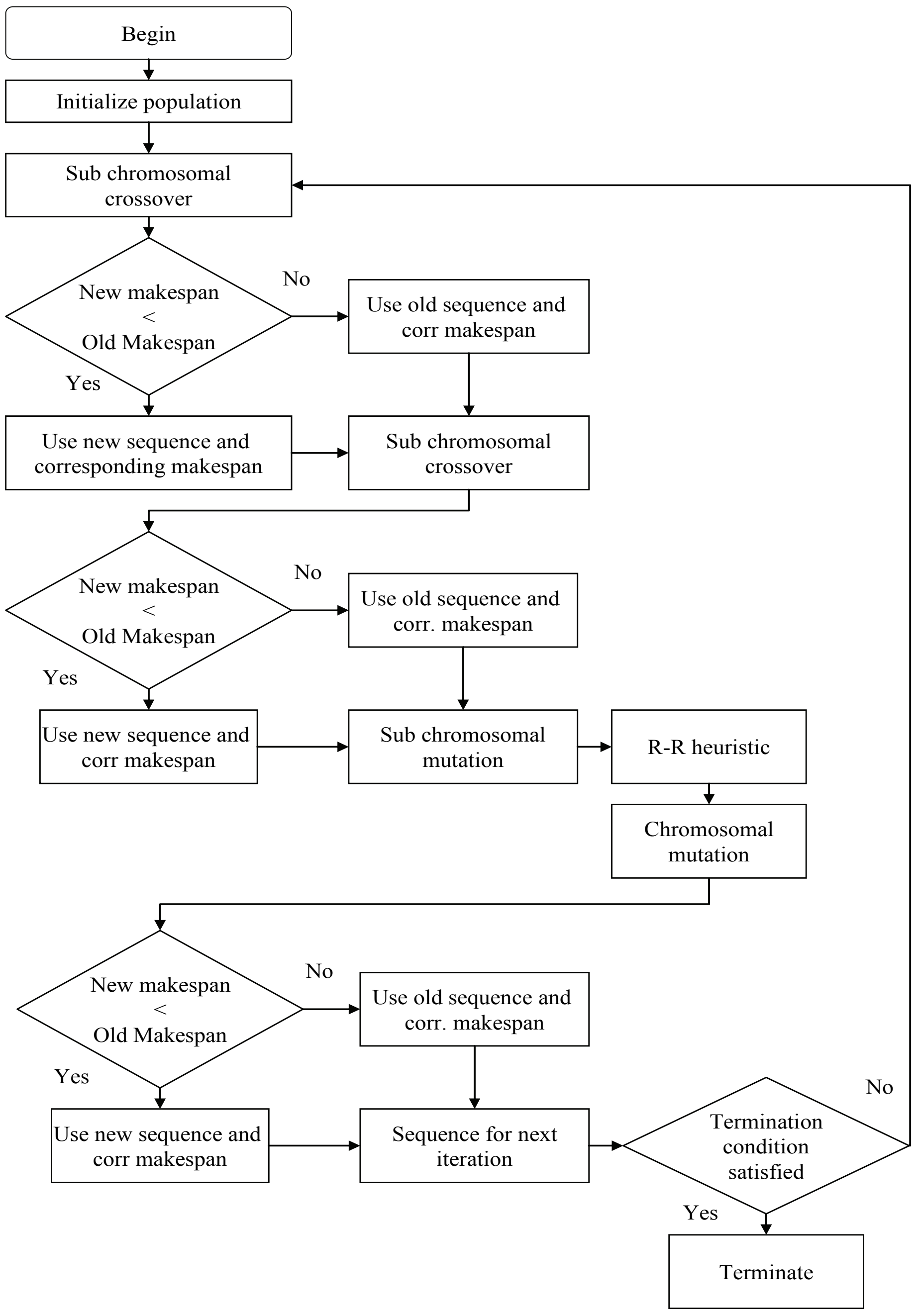
Begin
Initialize population
Sub chromosomal crossover
New makespan < Old Makespan
No
Yes
Use old sequence and corr makespan
Use new sequence and corresponding makespan
Sub chromosomal crossover
New makespan < Old Makespan
No
Yes
Use old sequence and corr. makespan
Use new sequence and corr makespan
Sub chromosomal mutation
R-R heuristic
Chromosomal mutation
New makespan < Old Makespan
No
Yes
Use old sequence and corr. makespan
Use new sequence and corr makespan
Sequence for next iteration
Termination condition satisfied
No
Yes
Terminate

Fig. 1. Flow chart of ISFHA

4.1.3. Problem Formulation

A mathematical model based on the above said JSSP is built. For the model built, the notations are defined as given below:

n	number of jobs
m	number of operations for one job
O_i	completed time of operation i ($i = 0, 1, 2, \ldots, n \times m + 1$)
t_i	processing time of operation i on a given machine
ω_{im}	the flag of operation i initiates machine m
P_i	all predecessor operations of operation i
$A(t)$	the set of operations being processed at time t
o_{ji}'	ith operation of job j
C_{max}	makespan

According to the notations, the conceptual model of JSSP can be defined as follows:

$$\text{Minimize } O_{n \times m + 1} \tag{2}$$

$$O_q \le O_i - t_i \,,\; i = 0, 1, 2 \ldots n \times m + 1;\; q \in P_i \tag{3}$$

$$\sum_i \omega_{im} \le 1 \,,\; m \in M,\;\; t \ge 0,\; i \in A(t) \tag{4}$$

$$O_i \ge 0,\; i = 0, 1, 2\ldots.., n \times m + 1 \tag{5}$$

The objective function in Eq. (2) is to minimize makespan i.e the completion time of the last operation. The constraint of precedence relationship is defined by Eq. (3). Eq. (4) indicates that one machine can process at most one operation at a time. Eq. (5) indicates that the finish time must be positive.

4.2. Flow shop scheduling problem

4.2.1. Problem Description

The objective of the FSSP is to find an appropriate permutation schedule for jobs that minimizes the maximum completion time. Every job has m operations. The flow-shop has m machines. Every operation of any job must follow the same machine sequence M_1, M_2,. . ., M_m. That is, the rth operation of job i is performed by machine M_r with fixed processing time $T(r, i)$, $1 \le r \le m$, and $1 \le i \le k$.

4.2.2. Assumptions

The FSSP is considered under the following basic assumptions:

- Every machine can execute only an operation at the same time.
- All machines execute jobs in the order of the pre-defined permutation schedule.
- Machine breakdown does not occur, which means all the machines are continuously available throughout the production stage.
- Pre-emption of operations is not allowed, which means the processing of an operation cannot be interrupted once started.

4.2.3. Problem Formulation

The equation of computing the maximum completion time of the permutation schedule is given as follows:

$$C(1, 1) = T(1, 1) \tag{6}$$

$$C(1, i) = C(1, i\text{-}1) + T(1, i) \quad (7)$$
$$C(r, 1) = C(r\text{-}1, 1) + T(r,1) \quad (8)$$
$$C(r, \mathrm{i}) = \max(C(r, i\text{-}1), C(\mathrm{r}\text{-}1, i)) + T(r, i) \quad (9)$$

In Eq. (9), $T(r, i)$ means the execution time of the r^{th} operation of the i^{th} job on machine M_r, $C(r, i)$ means the maximum completion time of the *i*th job on machine *Mr*, $1 \leq r \leq m$, and $1 \leq i \leq k$. If *m* is equal to the total number of machines, and *k* is equal to the total number of jobs, then $C(m, k)$ means makespan.

5. Implementation of ISFHA

The algorithm consists of the following steps:

1. Generation of initial population
2. Sub chromosomal level crossover
3. Sub chromosomal level mutation (inverse and single point mutation)
4. Robust-Replace heuristic
5. Chromosomal level mutation

5.1 Generation of initial population

A set of initial sequences are randomly generated according to the problem size and the initial makespan values are calculated. Encoding a schedule to a search solution is a key issue for ISFHA, which should be well considered because of the constraints of scheduling problems, especially for JSSP (Cheng et al., 1996).

5.2 Sub chromosomal level crossover

The initial chromosomes are modified by sub chromosomal crossover. The entire string length is split into sub chromosomes of equal length and these sub chromosomes are moved randomly to form a new chromosome. If the makespan corresponding to this new chromosome is less than the makespan of the parent chromosome, the parent chromosome is replaced by the new chromosome. Otherwise, the parent chromosome is retained.

Parent Chromosome

7 4 1	12 8 9	5 2 13	10 3 11	6
1	2	3	4	5

Sub chromosome length = 3

After Sub Chromosomal Crossover

6	5 2 13	10 3 11	12 8 9	7 4 1
5	3	4	2	1

5.3 Sub chromosomal level mutation

5.3.1 Inverse mutation

In a sequence, two positions *i* and *j* are randomly selected. The portion of the sequence between these two positions is inverted to get a new mutated sequence. The new sequence represents the sequence of operations after mutation. If the makespan of the mutated sequence is less than the makespan of the original sequence, the old sequence is replaced by the new sequence.

Original sequence

4 5 1 12 10 2 11 13 7 8 9 6 3

Mutated sequence

4 **6** **9** **8** **7** **13** **11** **2** **10** **12** **1** **5** 3

Mutation between positions 2 and 12

5.3.2 Single point mutation

A random operation is selected in the sequence and moved to another random position in the sequence. If the makespan of the resulting sequence is less than that of the previous one, it replaces the previous sequence.

Original sequence

10 7 12 1 2 **4** 13 8 5 3 11 9 6

Mutation of operation at position 6 to position 10

Mutated sequence

10 7 12 1 2 13 8 5 3 **4** 11 9 6

5.4 Robust-replace heuristic

Consider an instance of implementation with ten random sequences and their corresponding makespan and robust factors as given in Table 1. The robust factor for each of the schedules is calculated using the formula given in Eq. (1).

The average value of the robust factor for the set of ten sequences is calculated. The sequences having robust factor values below the average are selected and replaced by new schedules. The retained schedules are given in Table 2.

Table 1
Initial sequences

Sequence													Makespan	Robust Factor
1	2	7	4	12	3	8	5	10	6	11	9	13	159.42	0.006273
7	1	10	12	8	4	2	13	3	11	5	9	6	147.11	0.006797
7	1	10	2	11	12	4	8	3	13	5	9	6	155.71	0.006422
4	5	10	11	12	1	2	7	8	9	13	6	3	145.01	0.006896
7	4	1	12	8	9	2	10	5	13	6	11	3	147.73	0.006769
4	7	8	5	10	6	11	1	2	9	12	3	13	149.57	0.006686
10	7	12	1	2	4	13	8	5	3	11	9	6	153.44	0.006517
4	5	7	6	8	1	10	2	9	3	11	12	13	147.75	0.006768
7	1	4	12	2	8	10	9	5	13	6	11	3	146.00	0.006849
4	5	1	6	12	13	7	10	11	2	3	8	9	148.65	0.006727

Average robust factor = 0.006670

Table 2
Sequences retained

Sequence													Makespan	Robust Factor
7	1	10	12	8	4	2	13	3	11	5	9	6	147.11	0.006797
4	5	10	11	12	1	2	7	8	9	13	6	3	145.01	0.006896
7	4	1	12	8	9	2	10	5	13	6	11	3	147.73	0.006769
4	7	8	5	10	6	11	1	2	9	12	3	13	149.57	0.006686
4	5	7	6	8	1	10	2	9	3	11	12	13	147.75	0.006768
7	1	4	12	2	8	10	9	5	13	6	11	3	146.00	0.006849
4	5	1	6	12	13	7	10	11	2	3	8	9	148.65	0.006727

5.5 Chromosomal level mutation

The two mutation processes are carried out among the chromosomes in the population while the sub chromosomal level mutation is restricted within a single chromosome. The mutation process includes inverse mutation and single point mutation as given in Sec. 5.3.

5.6 Selection of parameters

The ISFHA has the following parameters to be defined.

(1) Size of population (S)
(2) No. of generations for which the algorithm is applied (N)
(3) Length of sub chromosome (L_s)
(4) Probability of sub chromosomal level crossover (P_c)
(5) Probability of inverse mutation (P_i)
(6) Probability of single point mutation (P_s)

An analysis is performed by assigning a set of values to each of the above parameters. The performance of the algorithm under each set of parameters is examined to arrive at an appropriate set of parameters that will provide most favorable solutions to the problems considered.

5.6.1. Population size and number of generations

First, the influences of population size and generation number on the near-optimal solutions were observed using an initial set of experiments. Table 3 shows the result obtained on this experiment. From the table, we learn that S = 10 and N = 1000 offered the best results for this experiment. Although a larger population size and generation number can definitely result in better performance, overall it is still reasonable to expect that those two parameters can be set to higher values to get positive results for later larger experiments.

Table 3
Analysis of N and S

Population Size	Generation Number	Makespan value
5	500	934
10	1000	927
15	1000	931
15	2000	929
20	2000	930
20	3000	931

5.6.2. Sub chromosomal crossover parameters

The two parameters, which decide the sub chromosomal level crossover operations are the length of the sub chromosome and the probability of sub chromosomal crossover taking place. The length of the sub chromosome decides on the quality of local search taking place. Probability of the sub chromosomal crossover is kept low so that not too many unfit sequences are produced.

5.6.3. Mutation probabilities

Mutation serves as a limited local search and maintains diversity in the population, but it is difficult to control its behavior and to choose a mutation rate. Both the mutation process acts on the current population simultaneously so that they can explore a wide range of the solution space to obtain high efficiency and good quality.

From Table 4, it shows that small mutation probabilities will produce better results than larger ones. Because the purpose of mutation is to prevent the pre-maturing of genetic algorithms, higher mutation probabilities may produce a greater percentage of ‘‘unfit” offspring, even when heuristics are used to improve their likelihood of survival. Since the scheduling problem has many restrictions, a high mutation rate would indeed produce a high proportion of such ‘‘doomed” offspring (Beyer et al., 2002; Rees & Koehler, 2006).

In the experiments reported herein, mutation probabilities between 0.001 and 0.005 produced the best results. Accordingly, the default inverse mutation probability was taken to be 0.001 and single mutation probability was taken to be 0.005 for the remainder of the work. For single point mutation, the probability is slightly higher than inverse mutation since the mutation is self induced.

Table 4
Analysis of P_i and P_s

Probability		Makespan
Inverse Mutation	Single point mutation	
0.1	0.1	941
0.05	0.1	937
0.01	0.01	930
0.001	0.01	931
0.001	0.001	935
0.001	0.005	928

5.6.4. Results of parameter analysis

Table 5 shows the results of the overall analysis done on the parameters.

Table 5
Parameter analysis

Parameter	Values taken for analysis	Best value	Corresponding best makespan
S	5, 10, 15, 20	10	927
N	500, 1000, 2000, 3000	1000	927
L_s	3, 4, 5	3	928
P_c	0.1, 0.2, 0.3, 0.4, 0.5	0.4	930
P_i	0.1, 0.05, 0.01, 0.001	0.001	928
P_s	0.1, 0.01, 0.001, 0.005	0.005	928

6. Experimental Results and Discussion

6.1 Job shop scheduling

6.1.1 Experimental setup

Twenty instances are taken from the OR-Library (Beasley, 1990) as benchmarks to test the proposed algorithm. In the 20 instances, FT06 and FT10 were designed by Fisher and Thompson (Fisher and Thompson, 1963) and instances LA21–LA38 were designed by Lawrence (Lawrence, 1984). The schedules for each problem are generated using ISFHA. The results obtained by ISFHA are compared with the Best Known Solution, GA (Goncalves et al., 2005), Modified PSO (Lin et al., 2010) and CSFHA (Vikram & Chandramouli, 2011). The coding for the optimization of scheduling has been developed using MATLAB. The Percentage Relative Deviation (PRD) from the Best Known Solution is given by Eq. (10).

$$\text{Percentage relative deviation} = \frac{\text{Optimal Solution} - \text{Best Known Solution}}{\text{Best Known Solution}} \times 100 \tag{10}$$

6.1.2 Results and Discussion

The results in the Table 6 demonstrate the PRD from the Best Known Solution obtained using GA, Modified PSO, CSFHA and ISFHA for 20 instances: FT06 and FT10 and LA21-LA38. For all the problem instances, the ISFHA performs exceedingly well.

Table 6
Results for JSSP

Problem	$n \times m$	GA	MPSO	CSFHA	ISFHA
FT06	6 × 6	0.00	0.00	0.00	0.00
FT10	10 × 10	0.00	0.00	0.00	0.00
LA21	15 × 10	0.00	0.00	0.00	0.00
LA22	15 × 10	0.86	0.54	0.11	0.05
LA23	15 × 10	0.00	0.00	0.00	0.00
LA24	15 × 10	1.93	0.64	0.64	0.57
LA25	15 × 10	0.92	0.00	0.31	0.00
LA26	20 × 10	0.00	0.00	0.00	0.00
LA27	20 × 10	1.70	0.32	1.54	0.22
LA28	20 × 10	1.32	0.00	1.07	0.00
LA29	20 × 10	3.82	1.82	0.95	0.82
LA30	20 × 10	0.00	0.00	0.00	0.00
LA31	30 × 10	0.00	0.00	0.00	0.00
LA32	30 × 10	0.00	0.00	0.00	0.00
LA33	30 × 10	0.00	0.00	0.00	0.00
LA34	30 × 10	0.00	0.00	0.00	0.00
LA35	30 × 10	0.00	0.00	0.00	0.00
LA36	15 × 15	0.87	0.79	0.00	0.00
LA37	15 × 15	0.79	1.00	0.36	0.20
LA38	15 × 15	1.92	1.00	0.50	0.36
	Mean	0.71	0.31	0.27	0.11

It can be seen from Table 6 that the ISFHA produces an overall average PRD (APRD) value equal to 0.11%, which is substantially lower than that of GA (0.71%), MPSO (0.31%) and CSFHA (0.27%). For all the problem sizes, the proposed ISFHA generates the lowest PRD values.

The ISFHA outperforms the existing CSFHA, GA and MPSO in terms of the overall APRD for the job shop problems with makespan criterion. Fig. 2 shows the difference in convergence of CSFHA and ISFHA for problem LA22. Fig. 2 clearly shows ISFHA achieves a better solution faster than CSFHA for JSSP.

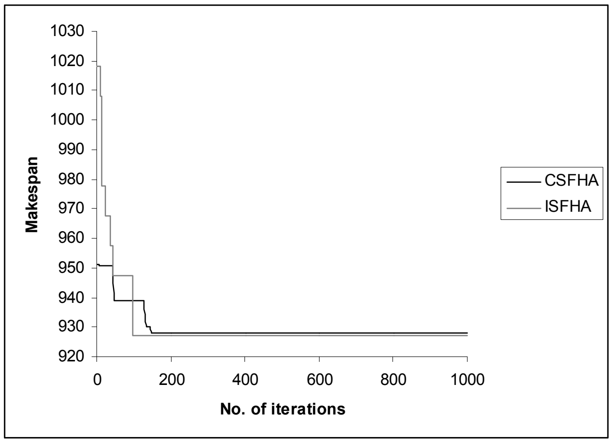

Fig. 2. Convergence graph of CSFHA and ISFHA for JSSP

6.2 Flow Shop Scheduling

6.2.1 Experimental Setup

To test the performance of the proposed algorithm, an extensive experimental evaluation and comparison with other recent existing methods are provided based on the well-known flow shop benchmark set of Taillard (1990), which is composed of 12 groups of the given problems with the size ranging from 20 jobs and 5 machines to 500 jobs and 20 machines, and each group consists of 10 instances. For each instance the algorithm is run with five independent replications, and in each replication the PRD is calculated as given in Eq. (9).

6.2.2 Results and Discussion

The results of the PRD obtained from ISFHA are compared with results obtained from Hybrid Genetic Algorithm (HGA) (Wang et al., 2006), hybrid alternate two phases PSO (Changsheng Zhang et al., 2010), hybrid modified global-best harmony search algorithm (I-Hong Kuo et al., 2009) and CSFHA applied for the flow shop scheduling problem. The results are shown in Table 7.

Table 7
Results for FSSP

Problem	$n \times m$	HGA	ATPPSO	hmGHS	CSFHA	ISFHA
Ta010	20 × 5	1.66	0.21	0.12	0.10	0.07
Ta020	20 × 10	1.48	1.08	0.08	0.08	0.04
Ta030	20 × 20	1.16	0.71	0.06	0.06	0.05
Ta040	50 × 5	4.37	0.02	0.39	0.31	0.29
Ta050	50 × 10	5.34	2.97	0.52	0.47	0.38
Ta060	50 × 20	4.96	3.91	0.25	0.24	0.24
Ta070	100 × 5	6.32	0.43	1.17	1.19	1.01
Ta080	100 × 10	5.63	0.94	0.41	0.43	0.43
Ta090	100 × 20	4.13	3.98	0.41	0.38	0.36
Ta100	200 × 10	4.94	1.60	0.35	0.25	0.22
Ta110	200 × 20	3.56	4.38	0.20	0.07	0.09
Ta120	500 × 20	2.97	2.98	0.15	0.13	0.13
	Mean	3.88	1.93	0.34	0.31	0.28

It can be seen from Table 7 that the ISFHA produces an overall APRD value equal to 0.28%, which is substantially lower than that of HGA (3.88%), ATPPSO (1.93%), hmGHS (0.34%) and CSFHA (0.31%). For 11 out of 12 problem sizes, the proposed ISFHA generates the lowest PRD values. The ISFHA outperforms the existing HGA, ATPPSO, hmGHS algorithms and CSFHA in terms of the overall APRD for the flow shop problems with makespan criterion. Fig. 3 shows the difference in convergence of CSFHA and ISFHA for problem TA001. Fig. 3 clearly shows ISFHA achieves a better solution faster than CSFHA for FSSP.

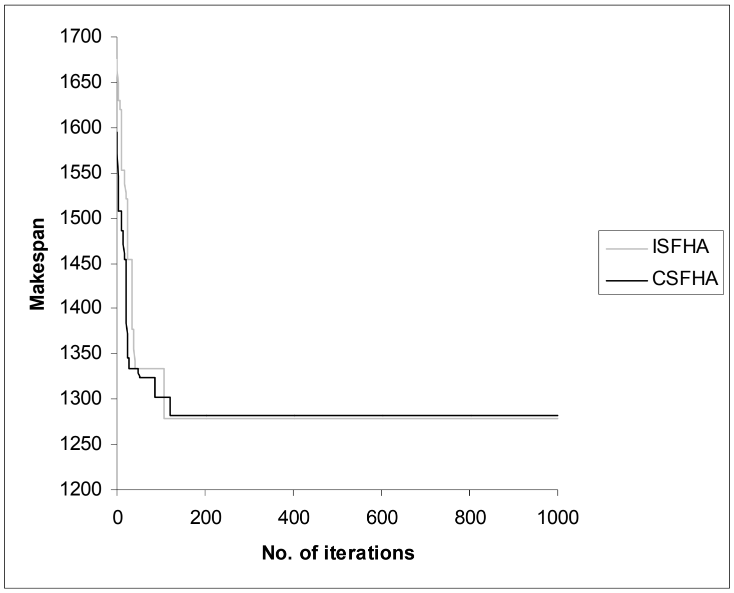

Fig. 3. Convergence graph of CSFHA and ISFHA for FSSP

7. Conclusion

In this paper, a new improved sheep flock heredity algorithm named ISFHA is proposed to solve extensive scheduling problems that include job shop scheduling and flow shop scheduling.

One of the main contributions is to prove that ISFHA can reach the favorable solution area in the search space quickly with smaller population size and faster convergence than CSFHA. Another contribution is to prove that the superiority of solution for the scheduling problems found by the ISFHA. It is found to be better than those obtained by GA, PSO and its variations, hybrid harmony search algorithm and CSFHA as evident from Table 6 and Table 7. The last contribution is to prove that both the local searching ability and global searching ability of the ISFHA are enhanced by means of single point mutation and R-R heuristic.

The future research includes application of ISFHA to combinatorial optimization problems. The algorithm can be used for single or multiple objectives case considering different criteria like mean flow time, total tardiness, and maximum tardiness. Furthermore, it is possible that the proposed algorithm can be applied to the scheduling problems in various manufacturing systems such as cellular manufacturing and flexible manufacturing. This algorithm can also be applied to optimize multi machine environment and the dynamic JSSP. Another development can be made by combining this algorithm with other available search algorithms to provide good results.

References

Araújo, D.C. & Nagano, M.S. (2011). A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem. *International Journal of Industrial Engineering Computations*, 2, 155-166.

Arun Vikram, M. S. & Chandramouli, A. (2011). An efficient evolutionary approach to compute optimal schedules for the Job Shop Scheduling Problem. *Proceedings of National Conference on Innovations in Mechanical Engineering*, 1, 218-224.

Baker, K. R. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Beasley, J.E. (1990). OR-library: Distributing test problems by electronic mail, Journal of Operations Research Society, 41(11), 1069–1072.

Beyer, H.G., Schwefel, H.P. & Wegener, I. (2002). How to analyze evolutionary algorithms. Technical Report No. CI-139/02, University of Dortmund.

Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T., & Nickolas, S. (2005). Solving job shop scheduling problems using artificial immune system. *International Journal of Advanced Manufacturing Technology,* 31(5-6), 580-593.

Cheng, R., Gen, M. & Tsujimura, Y. (1996). A tutorial survey of jobshop scheduling problems using genetic algorithms – I. Representation. *Computers and Industrial Engineering*, 30, 983–997.

Fink, A., & Vob, S. (2003) Solving the continuous flow shop scheduling problem by metaheuristic. *European Journal of Operational Research*, 151, 400–414.

Fisher, H. & Thompson, G. L. (1963). *Industrial scheduling*. Englewood Cliffs, NJ: Prentice-Hall.

French, S. (1982). *Sequencing and scheduling: An introduction to the mathematics of the job shop*. Chichester, West Sussex, E. Horwood.

Garey, M., Johnson, D., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1, 117–129.

Giffler, B., & Thompson, G. L. (1960). Algorithms for solving production scheduling problems. *Operations Research*, 8(4), 487–503.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison Wesley.

Goncalves, J.F., Mendes, J.J.D.M. & Resende, M.G.C. (2005). A hybrid genetic algorithm for the jobshop scheduling problem. *European Journal of Operational Research*, 167, 77–95.

Grefenstette, J.J. (1988). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1), 122–128.

Kuo, I.H., Horng, S.H., Kao, T.W., Lin, T.L., Lee, C.L., Terano, T., & Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on hybrid particle swarm optimization model. *Expert Systems with Applications*, 36, 7027-7032.

Koichi Nara, Tomomi Takeyama & Hyunchul Kim. (1999). A New Evolutionary Algorithm Based on Sheep Flocks Heredity Model and Its Application to Scheduling Problem. *IEEE Transactions*, VI, 503-508.

Lawrence, S. (1984). Resource constraint project scheduling: An experimental investigation of heuristic scheduling techniques. Graduate School of Ind. Admin.. Carnegie Mellon University, Tech. Rep.

Leung, Y., Gao, Y. & Xu, Z.B. (1997). Degree of population diversity – a perspective on premature convergence in genetic algorithms and its markov-chain analysis. *IEEE Transactions on Neural Networks*, 8(5), 1165–1176.

Lin, T.L., Horng, S.J. Kao, T.W., Chen, Y.H., Run, R.S., Chen, R.J., Lai, J.L. & Kuo, I.H. (2010), An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37, 2629–2636.

Rees, J. & Koehler, G. J., (2006). Learning genetic algorithm parameters using hidden Markov models. *European Journal of Operational Research,* 175 (2), 806–820.

Rudolph, G. (1994). Convergence properties of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1), 96– 101.

Rui Zhang & Cheng Wu. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, 10, 79–89.

Sayadi, M.K., Ramezanian, R. & Ghaffari-Nasab, N. (2010). A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations*, 1, 1-10.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problems. *European Journal of Operational Research*, 47, 65–74.

Wang, L., Zhang, L., & Zheng, D. Z. (2006). An effective genetic algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research*, 33, 2960–2971.

Wang, L. & Tang, D.B. (2011). An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem. *Expert Systems with Applications,* 38, 7243–7250.

Wang, L., Pan, Q.K. & Tasgetiren, F. M. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers and Industrial Engineering*, 66(1), 76-83.

Wiers, V. C. S. (1997). A review of the applicability of OR and AI scheduling techniques in practice. *Omega- International Journal of Management Science*, 25(2), 145-153.

Yang, R. & Donglas, I. (1998). Simple genetic algorithm with local turning-efficient global optimizing technique. *Journal of Optimization Theory and Applications*, 98(2), 449–465.

Zhang, C., Ning, J. & Ouyang, D. (2010). A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. *Computers and Industrial Engineering*, 58, 1-11.