

Full Length Research Paper

An efficient job shop scheduling algorithm based on artificial bee colony

Minghao Yin^{1,2}, Xiangtao Li^{1*} and Junping Zhou¹

¹College of Computer Science, Northeast Normal University, Changchun, 130117, P. R. China.

²Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, 130012, Changchun, P. R., China.

Accepted 24 November, 2010

The job shop scheduling problem (JSSP) is an NP-hard problem of wide engineering and theoretical background. In this paper, a discrete artificial bee colony based memetic algorithm, named DABC, is proposed for solving JSSP. Firstly, to make artificial bee colony (ABC) suitable for solving JSSP, we present a food source as a discrete job permutation and use the discrete operation to generate a new neighborhood food source for employing a bee colony, an onlooker bee colony and a scout bee colony. Secondly, three mutation operations are proposed to make DABC applicable for the job shop scheduling problem. Thirdly, the fast local search is used to enhance the individuals with a certain probability. Fourthly, the pairwise based local search is used to enhance the global optimal solution and help the algorithm to escape from the local minimum. Additionally, simulations and comparisons based on JSSP benchmarks are carried out, which show that our algorithm is both effective and efficient.

Key words: Artificial bee colony, job shop scheduling, memetic algorithm, local search.

INTRODUCTION

Scheduling problems play an important role in both manufacturing systems and industrial process for improving the utilization of resources, and therefore it is crucial to develop efficient scheduling technologies (Stadtler, 2005). The Job shop scheduling problem, one of the best known production scheduling problems, has been proved to be non-deterministic-polynomial-time – hard (NP) (Garey et al., 1976). Therefore, it is difficult to find a better algorithm to generate a solution within a reasonable time especially for the large-sized scheduling problems. Due to its significance in both academic and

engineering applications, several kinds of approaches have been proposed to solve JSSP and obtained some achievements.

In recent years, many researchers have made use of meta-heuristics algorithms to solve the job shop scheduling problem, such as genetic algorithm (Liu et al., 2006; Hajri et al., 2000; Gang and Wu, 2004; Amirthagadeswaran and Arunachalam, 2006), tabu algorithm (Geyik and Cedimoglu, 2004; Watson et al., 2003; Ponnambalam et al., 2000), simulated annealing (Low et al., 2004; Wang and Wu, 2000; Kolonko, 1999; Suresh and Mohanasundaram, 2006), particle swarm optimization (Lian et al., 2006; Xia and Wu, 2006), colony algorithm (Zhang et al., 2006), artificial neural network (Yu and liang, 2001; Yang and Wang, 2001; 2000),

*Corresponding author. E-mail: lixt314@gmail.com.

artificial immune system (Coello et al., 2003). The meta-heuristics algorithm usually can obtain fairly satisfactory solutions, while the solution processes are always time-consuming and vary dramatically according to their parameters and structure. In addition, it has become evident that the concentration on a sole meta-heuristic has some limitations.

Rather recently, researchers have found that a skilled combination of two or more meta-heuristic techniques, as called hybrid heuristics, can improve the performance especially when dealing with real-world and large scale problems. Although there is no free lunch (David and William, 1997; Yin et al., 2010) for the hybrid method, it is believed that by combining the advantages of different methods in a complementary way, we can have more efficient approaches. Therefore, we can find the hybrid particle swarm optimization (Lin et al., 2010), hybrid artificial immune algorithm (Ge et al., 2008), hybrid genetic algorithm (Goncalves et al., 2005) and other meta-heuristic algorithms (Dauzere and Paulli, 1997) for the job shop scheduling.

Among the existing meta-heuristic algorithms, an evolution technique, the artificial bee colony algorithm is a new population-based heuristic evolutionary algorithm that is simple to be implemented and has little or no parameters to be tuned (Karaboga and Basturk, 2007). This approach is inspired by the intelligent foraging behaviour of honeybee swarm. Up till now, most published works on ABC mainly focused on solving the complex continuous optimization problem (Karaboga and Basturk, 2008; Karaboga and Ozturk, 2010). Within our knowledge, only few of researchers have used the ABC algorithm to solve the scheduling problems. In 2010, (Pan et al., 2010) proposed a discrete artificial bee colony to solve the lot-streaming flow shop scheduling with the criterion of total weighted earliness and tardiness penalties under both idling and no-idling cases. However, this field of study is still in its early days; a large number of future researches are necessary in order to develop those algorithms based on ABC based for solving JSSP other than only for those areas the inventors originally focused on.

In this paper, we propose a discrete artificial bee colony algorithm with some local search mechanisms containing fast local search and pairwise based local search for solving JSSP. The crucial idea behind DABC can be summarized as follows. Firstly, to make ABC be suitable to solve JSSP, we present a food source as a discrete job permutation and apply the discrete operation to generate a new neighborhood food source for three different bees. Secondly, three mutation operations are proposed to enable the DABC to solve the job shop scheduling problem. Thirdly, multiple different neighborhoods are designed and incorporated as a local search scheme into the searching process to enrich the searching behaviors and to avoid premature convergence.

Fast local search is proposed as a component of DABC, for the sake of improving DABC's local search performance. Fourthly, the pairwise-based local search is used to enhance the global optimal solution and to help the algorithm to escape from the local minimum.

The rest of this paper is organized as follows. In section 2 and 3, we introduce JSSP and ABC respectively. In Section 4, the DABC algorithm is proposed after presenting several important concepts. The experimental results of the DABC and comparisons to other previous algorithms are shown in section 5. In the last section we conclude this paper and point out some future work.

JOB SHOP SCHEDULING PROBLEM

The job shop scheduling problem (JSSP) in the paper consists of a set of jobs on a set of machines with the objective of minimizing the make span. In job shop scheduling, each machine can handle at most one job at a time. We consider one job consists of m operations that are to be processed in a specified machine, and once an operation initiates processing on a given machine, it must complete processing on that machine without interruption. It is sequence in which one job to be processed is predefined and maybe different from other jobs. The objective of the job shop scheduling is to find a scheduling sequence that can minimize the maximum completion of the jobs. A good schedule is one that minimizes idle time the machines take up.

In general, a $n \times m$ job-shop scheduling problem is formulated as follows. Let $J = \{1, 2, \dots, n\}$ denote the set of operations to be scheduled, $M = \{1, \dots, m\}$ represent the set of machines, and $O = \{0, 1, \dots, n \times m, n \times m + 1\}$ be the set of operations that is to be scheduled, where the operation 0 and $n \times m + 1$ are dummy, have no duration and denote the initial and last operations respectively. The operations are interrelated by two kinds of constraints. The precedence constraint is used to make each operation j to be scheduled after all predecessor operations P_j are finished. Moreover, operation j can only be scheduled if the machine it requires is idle. Further, Let d_j and F_j denote the fixed duration time and the finish time of operation j . Let $A(t)$ be the set of operations being processed at time t , and let $\omega_{jm} = 1$ if operation j requires machine m to be processed and $\omega_{jm} = 0$ otherwise.

According to the description listed above, its model is as follows:

$$\text{Minimize } F_{n \times m + 1} \quad (1)$$

Subject to:

$$F_k \leq F_j - d_j \quad ; \quad j=1, \dots, n \times m + 1 ; k \in P_j \quad (2)$$

$$\sum_{j \in A(t)} \omega_{jm} \leq 1, \quad m \in M, t \geq 0 \quad (3)$$

$$F_j \geq 0, \quad j=1, 2, \dots, n \times m + 1. \quad (4)$$

The objective function (1) minimizes the finish time of the last operation and namely the makespan. The precedence relations between operations are represented by constraint (2). Constraint (3) denotes that one machine can only process one operation at a time, and constraint (4) forces the finish times to be non negative.

ARTIFICIAL BEE COLONY

The artificial bee colony algorithm is an evolutionary algorithm first introduced by Karaboga in 2005. This algorithm simulates the foraging behavior of bee colony. In this algorithm, the model of the ABC algorithm consists of three groups of bees: employed bees, onlooker bees, and scout bees. The first half of the colony consists of the employed bees and the second half contains onlookers. For every food source, there is only one employed bee. In other words, the number of bees is equal to the number of food sources. The employed bee of an abandoned food source becomes a scout. Employed bees are responsible for exploiting the nectar sources explored before; share their information with onlookers within the hive and then the onlookers select one of the food sources. Onlookers select a food source within the neighborhood of the food source chosen by themselves. An employed bee of which the source has been abandoned becomes a scout and starts to search a new food source randomly (Karaboga and Akay, 2009). Main steps of the ABC algorithm simulating these behaviors are listed as below:

Begin
Initialize
Repeat

Step 1: move the employed bees onto their food sources and determine their nectars amount.

Step 2: Move the onlookers onto their food sources and determine their nectar amounts.

Step 3: move the scouts for searching for new food sources.

Step 4: memorize the best food source found so far.

UNTIL (Requirement are meant)

End.

Each cycle of the search consists of three steps: moving the employed and onlooker bees to the food sources, calculating their nectar amounts, determining the scout bees, and then moving them randomly onto the possible food source. A food source stands for a potential solution to the problem to be optimized. The ABC algorithm is an iterative algorithm. It starts by associating all employed bees with randomly generated food solutions. The initial population of solutions is filled with SN number of D dimensions generated randomly. Let $X_i = \{x_{i1}, x_{i2}, \dots, x_{iD}\}$ represent the i^{th} food source in the population, SN be the number of food source which is equal to the number of the employed bees or looker bees. Each food source is generated as follows:

$$x_{ij} = LB_j + (UB_j - LB_j) \times r \quad (5)$$

Where $i \in \{1, 2, \dots, SN\}$, $j \in \{1, 2, \dots, D\}$ are randomly chosen indexes, r is a uniform random number in the range $[0, 1]$, LB_j and UB_j are lower and upper bounds for the dimension j respectively.

Then each employed bee x_{ij} generates a new food source v_{ij} in the neighborhood of its present position as follows.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (6)$$

$$k = \text{int}(\text{rand} * SN) + 1;$$

Where $\phi_{ij} = (\text{rand} - 0.5) \times 2$, ϕ_{ij} is a uniformly distributed real random number within the range $[-1, 1]$, $i \in \{1, 2, \dots, SN\}$, $k \in \{1, 2, \dots, SN\}$ and $k \neq i$, $j \in \{1, 2, \dots, n\}$ are randomly chosen indexes. After producing the new solution v_i , this new solution will be evaluated and compared to the x_i . If the objective fitness of v_i is smaller than the fitness of x_i , v_i is accepted as a new basic solution. Otherwise x_i would be obtained. The employed bee exploited the better solution.

When all employed bees have finished this process, an onlooker bee can obtain the information of the food sources from all employed bees and choose a food source depending on the probability value associated with that food source, using the following expression:

$$p_i = 0.9 \times \frac{fitness_i}{\max(fitness_i)} + 0.1 \quad (7)$$

Where $fitness_i$ is the fitness value of the solution i evaluated by its employed bee, which is proportional to the maximal value of the food source in the position i . Obviously, when the maximal value of the food source decreases, the probability with the preferred source by an onlooker bee decreases proportionally. Then the onlooker bee produces a new source in selected food source site by equation (6). The new source will be evaluated and compared to the primary food solution; if the new source has a better nectar amount than the primary food solution, the new source will be accepted.

After all onlookers have finished this process, sources are checked whether they are to be abandoned. If the food source does not improve through determined number of the trails "limit", then the food source is abandoned by its employed bee and then it becomes a scout. The scout starts to search for a food source randomly as follow:

$$x_{ij} = LB_j + (UB_j - LB_j) \times r \quad (8)$$

Where r is a uniform random number in the range $[0, 1]$.

After the new source is produced, another iteration of ABC algorithm begins. The whole process repeats again till the termination condition is met.

DISCRETE ARTIFICIAL BEE COLONY FOR JSSP

Solution representation

Because standard ABC is a continuous optimization algorithm, the standard continuous encoding scheme of ABC cannot be used to solve JSSP directly. In order to apply ABC to JSSP, one of the key issues is to construct a direct relationship between the job sequence and the vector of individuals in ABC. Therefore, in this paper, we propose a discrete artificial bee colony algorithm. In general, the food source x to the problem of scheduling n independent jobs on the job shop is associated with a job sequence $\pi = (\pi_1, \pi_2, \dots, \pi_{n \times m})$ where π_k represents a job index, $1 \leq \pi_k \leq n$. A job represents a set of operations that have to be scheduled on m machines. In this formulation, each job number occurs m times in the permutation. We can scan the schedule sequence

$\pi = (\pi_1, \pi_2, \dots, \pi_{n \times m})$ from left to right, and then the i^{th} number of the schedule sequence is corresponding to the i^{th} operation of a job. An example, a food source in a 3×3 job shop scheduling problem is shown as follows.

In a 3×3 job shop scheduling problem, a permutation sequence is given as (3, 2, 2, 1, 1, 2, 3, 1, 3), in which each food source denotes one operation. Each job consists of three operations, and the job number is repeated three times. From the permutation sequence (3, 2, 2, 1, 1, 2, 3, 1, 3), the first 3 means the first operation of job 3, corresponding to o_{31} . The seven position denotes the second operation of job 3, corresponding to o_{32} . According to the scanning process, the sequence (2, 2) refers to (o_{21}, o_{22}) , and (1, 1) is corresponding to (o_{11}, o_{12}) . The next partial series (2, 3, 1) is corresponding to (o_{23}, o_{32}, o_{13}) . In this way, the permutation sequence (3, 2, 2, 1, 1, 2, 3, 1, 3) is corresponding to an operation sequence $(o_{31}, o_{21}, o_{22}, o_{11}, o_{12}, o_{23}, o_{32}, o_{13}, o_{33})$. This coding method can create an active schedule at every time. The prominent advantage is that each food source represents a feasible solution. And then this method is easily to proceed to the local searching which will be introduced in the next section including the fast local search and pairwise local search.

Employed bee colony

In the basic artificial bee colony algorithm, every employed bee determines a food source in the neighborhood of its currently associated food source and evaluates its nectar fitness. We know that each employed bee x_{ij} generates a new food source v_{ij} in the neighborhood of its present position as follows.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$$

$$k = \text{int}(\text{rand} * SN) + 1;$$

But this method cannot be used to a discrete job permutation directly. Therefore, the most important issue in applying ABC successfully to JSSP is to develop an effective problem mapping and solution generation mechanism. So, in this paper, we proposed a new updating mechanism.

We will propose three mutation methods for the ABC algorithm to solve the job shop problems; the details of these neighborhoods are presented in section 4.5. We use a local search based on insert operation to improve the employed bee colony for $n/2$ times. Using this method, the basic algorithm can solve the job shop scheduling problem. If the objective makespan of v_i is smaller than

the makespan of x_i , v_i is accepted as a new basic solution. Otherwise x_i would be obtained.

Onlooker bee colony

In the basic ABC algorithm, an onlooker bee can obtain the information of the food sources from all employed bees and choose a food source depending on the probability value associated with that food source, using the following expression:

$$p_i = 0.9 \times \frac{C_{\max}^i}{\max(C_{\max}^i)} + 0.1$$

Where C_{\max}^i is the minimum makespan which is evaluated by its employed bee. This is proportional to the maximal value of the food source in the position i . obviously, when the maximal makespan of the food source decreases, the probability with the preferred source by a looker bee decreases proportionally. The onlooker bee produces new food source using three mutation methods like that of the employed bee. We use a local search based swap operation to improve the onlooker bee colony for $n/2$ times. We will introduce it in section 4.5. The new source will be evaluated and compared to the primary food solution. If the new source has a better nectar amount than the primary food solution, the new source will be accepted.

Scout bee colony

In the ABC algorithm, if a solution does not improve for a predetermined number of trails "limit", then this food source is abandoned by its employed bee and then the employed bee becomes a scout. The scout produces a new permutation sequence randomly in the search scope. This new solution can carry new information for the population.

Mutation schemes

For JSSP, Four neighborhoods, *Insert*, *Adjacent Interchange*, *inverse* and *Swap* usually can be employed. In this paper, three neighborhoods, *Inverse*, *Insert* and *Swap*, are used to improve the diversity of population and enhance the quality of the solution. These three

neighborhood operations are shown in Figure 1. The details of these neighborhoods are as follows:

SWAP mutation: choose two different positions from a job permutation randomly and swap them.

INSERT mutation: choose two different positions from a job permutation randomly and insert the back one before the front.

INVERSE: inverse the subsequence between two different random positions of a job permutation.

SWAP mutation:

```
2 1 2 2 3 1 3 1
3 2 1 1 2 3 2 3 1
```

INSERT mutation

```
3 2 1 2 2 3 1 3 1
2 3 2 1 2 3 1 3 1
```

INVERSE mutation

```
3 2 1 2 2 3 1 3 1
3 2 1 1 3 2 2 3 1
```

Fast local search

Due to the parallel evolution framework of DABC, local search is easy to be incorporated for exploitation. In this section we present a fast local search which is embedded in DABC for solving JSSP. The purpose of the local search is to find a better solution from the neighborhood of a solution.

In order to enhance the local search ability and get a better solution, we propose a new fast local search to enhance the makespan of the food source. The algorithm is performed by using the above three mutation operations alternatively to avoid trapping into local optimal points sometimes.

A new individual enhancement scheme is proposed to combine the insert, swap and inverse operations. This method first selects an operation scheme from three mutation operations to operate an individual. The selected operation starts from an initial solution and attempts to move from the current solution x to its neighborhood x' .

If the objective fitness of x' is smaller than the fitness of the current solution, x' is accepted as a new basic solution. After finishing one scheme, the search process keeps generating the individual's neighborhood randomly

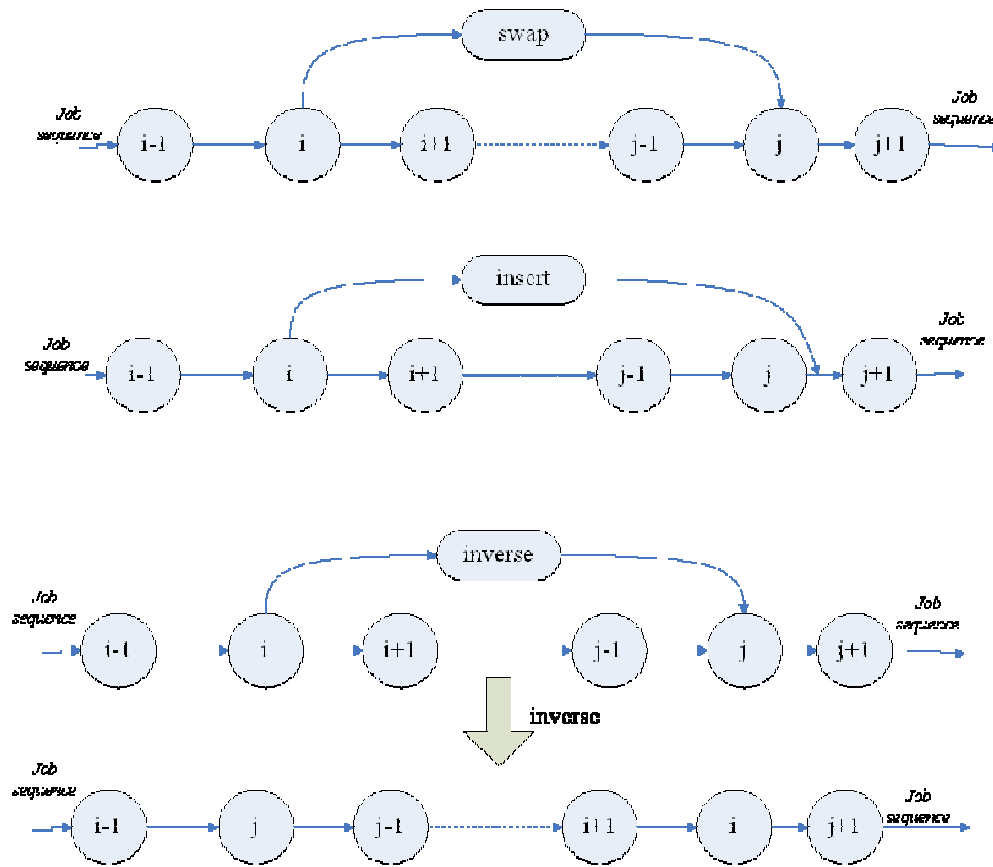


Figure 1. The neighborhood operations for local search.

and the solution is accepted until the stopping criterion is reached.

procedure fast local search

Begin

X_p the individual to be enhanced

The job permutation of the individual X_p is $\pi_p = [\pi_{p1}, \pi_{p2}, \dots, \pi_{pnm}]$. Evaluate fitness $f(\pi_p)$ for X_p individual.

for $i = 1$ to $nm \times (nm - 1)$

$q = \text{rand}()$;

Select $r1$ and $r2$ randomly, where $r1 \neq r2$

If $(0 \leq q \leq pr_s)$

Execute swapping scheme for the individual X_p , and obtain the individual X'_p

Else If $(pr_s \leq q \leq pr_s + pr_i)$

Execute inserting scheme for the individual X_p , and obtain the individual X'_p

Else $(pr_s + pr_i \leq q \leq pr_s + pr_i + pr_{inv})$

Execute inverting scheme for the individual X_p , and obtain the individual X'_p

End if

The job permutation of the individual X'_p is $\pi'_p = [\pi'_{p1}, \pi'_{p2}, \dots, \pi'_{pnm}]$. Evaluate fitness $f(\pi'_p)$ for X'_p individual.

If $(f(\pi'_p) - f(\pi_p) \leq 0)$

$X_p = X'_p$;

$f(\pi_p) = f(\pi'_p)$;

$\pi_p = \pi'_p$;

End if

End for

End.

Remarks

A partial algorithm about 3 type operations is listed in this algorithm. pr_s is the probability of executing swapping operation, pr_i is the probability of executing inserting operation, pr_{inv} is the probability of executing inverting operation.

Pairwise-based local search

In this section, a pairwise-based local search (Aldowaisan and Allahverdi, 2003) is employed for the global optimal solution. The crucial idea of pairwise-based local search is to swap the job orders of two adjacent jobs of an individual, and to compare the previous makespan and the new makespan. If the new makespan is lower than the previous makespan, we will accept the new individual. This method can also be regarded as a local search for DABC to enhance the global optimal solution for every generation. It examines each possible pairwise interchange of the job in the first position. Then

Table 1. The processing time.

Processing time			
Job	Operation		
	1	2	3
Job 1	2	1	2
Job 2	3	3	2
Job3	3	1	3

other positions are checked using the same operation. Whenever there is an improvement in the objective function, the orders of the jobs are interchanged. Pairwise-based search can be viewed as a detailed neighborhood searching process to enhance the exploitation ability.

The procedure of the pairwise -based local search algorithm for JSSP is as follows:

procedure Pairwise-based local search

Begin

X_{best} permutation of global best solution at iteration 1. $\pi_{best} = [\pi_{best1}, \pi_{best2}, \dots, \pi_{bestnm}]$

. Evaluate fitness $f(\pi_{best})$ for X_{best} individual.

$X_0 = X_{best}$

$\pi_0 = \pi_{best}$

$i=0$;

for $i=1$ to nm

$i=i+1$;

$j=0$;

for $j=1$ to nm

$j=j+1$;

Execute swapping the pair of jobs on position _{i} and position _{j} in the individual X_0 , and obtain the individual X_{new} .the job permutation is $\pi_{new} = [\pi_{new1}, \pi_{new2}, \dots, \pi_{newnm}]$

. Evaluate fitness $f(\pi_{new})$ for X_{new} individual.

If ($f(\pi_{new}) - f(\pi_{best}) < 0$)

$X_{best} = X_{new}$;

$f(\pi_{best}) = f(\pi_{new})$;

$X_0 = X_{new}$;

$\pi_{best} = \pi_{new}$;

End if

End for

End for

End.

DABC-based hybrid algorithm

This section discusses the structure of this hybrid algorithm. Figure 2 describes the framework of the proposed algorithm DABC. As is shown in Figure 2, there are mainly four strategies to update the individuals in the proposed algorithm. We present a food source as a discrete job permutation and apply the discrete operation

to generate new neighborhood food source for the three different bees. Three mutation operations are proposed to make the DABC solve the job shop scheduling. The fast-based local search is used to enhance the individuals with a certain probability, and therefore has a higher ability to approximate the optimal solution faster. The pairwise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum. We will use an example to describe the DABC. Tables 1 and 2 show the processing time and the initiated machine order of each operation. It is assumed that we want to improve the individual $\pi = (3,2,1,2,2,3,1,3,1)$, whose make span is 10 and the corresponding Gantt chart is shown in Figure 3. By using the proposed DABC algorithm, the final individual is $\pi = (1,2,3,1,3,1,2,3,2)$, whose make span is 9 and the corresponding Gantt chart is shown in Figure.4. The new individual is lower than the pervious individual. Thus, we will accept the new individual.

RESULTS AND DISCUSSION

To test the performance of the proposed DABC for the job shop scheduling, computational simulations are carried out with some well-studied problems taken from the OR-Library (Beasley, 1990). In this paper, 43 problems from two classes of JSSP test problems are selected. Instances FT06, FT10 and FT20 were designed by Fisher and Thomason (1963) and instances LA01-LA40 were designed by Lawrence (1984). So far, these problems have been widely used as benchmarks to certify the performance of algorithms by many researchers.

The DABC is coded in MATLAB 7.0, and in our simulation, numerical experiments are performed on a PC with Pentium 3.0 GHz Processor and 1.0 GB memory. Each instance is independently executed 15 times for DABC algorithm for comparison. The parameters used during the experimental process are defined in the following: SN=10, limit=20, the probability is 0.1, the maximum generation is 150. Numerical results are compared with those reported in some existing literature works using other algorithms (Ge et al., 2008; Gonçalves et al., 2005; Dauzere and Paulli, 1997; Ombuki and Ventresca, 2004; Aiex et al., 2003; Wang and zheng, 2002; Binato et al., 2001; Sabuncuoglu and Bayiz, 1999; Nuijten and Aarts, 1996; Nowicki and Smutnicki, 1996; Dorndorf and Pesch, 1995; Croce et al., 1995; Storer et

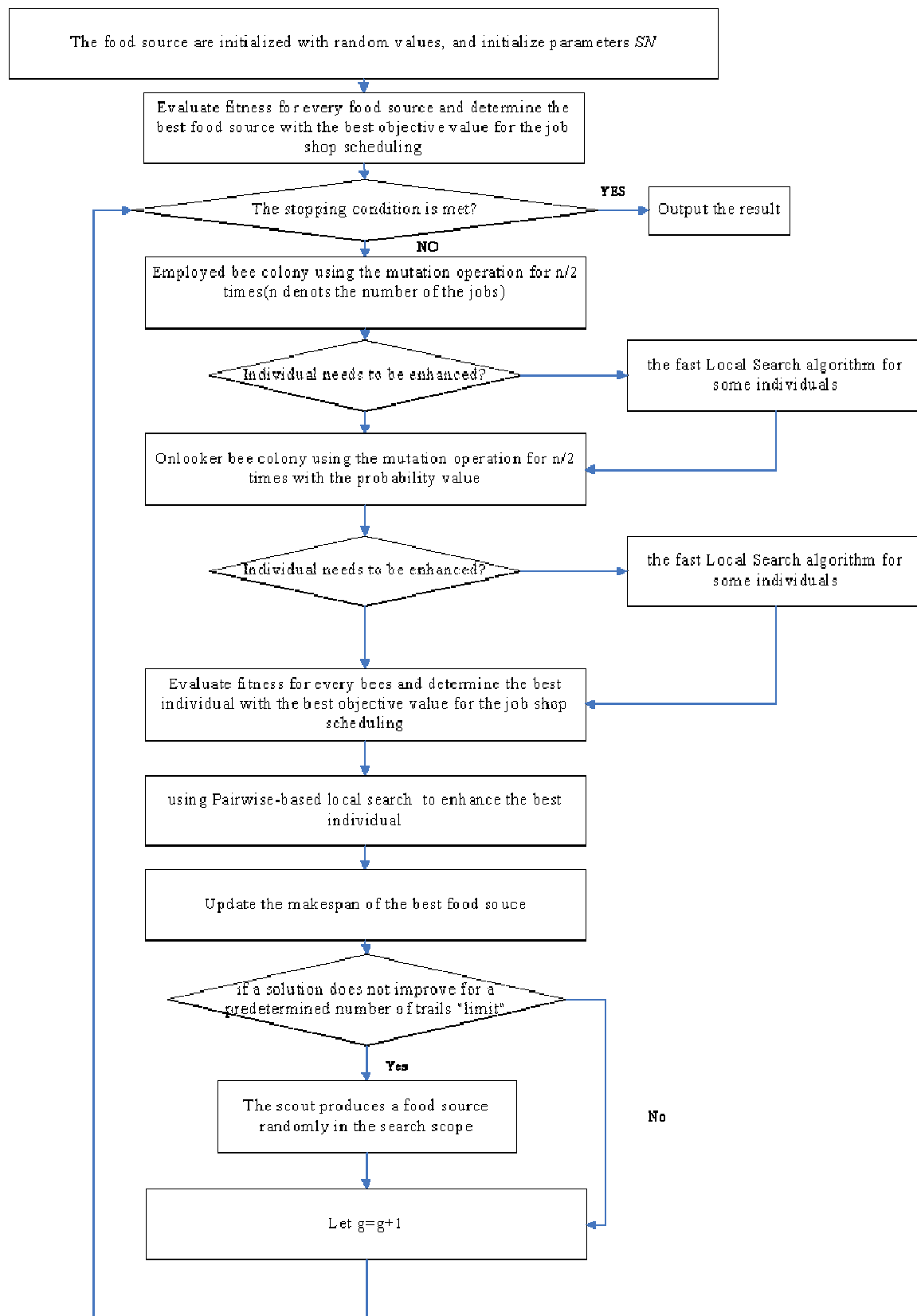
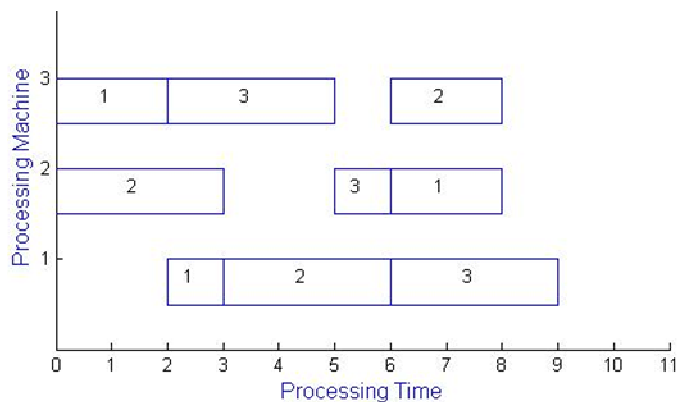
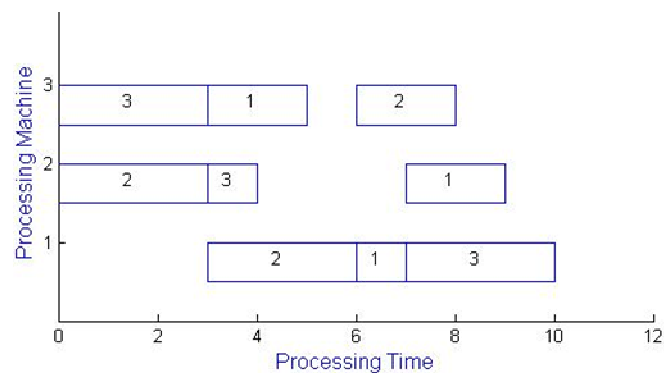


Figure 2. The framework of the proposed algorithm DABC for the job shop scheduling problem.

Table 2. The initiated machine order of each machine.

Job	Machine sequence		
	Operations		
	1	2	3
Job 1	M ₁	M ₃	M ₂
Job 2	M ₂	M ₃	M ₁
Job3	M ₁	M ₂	M ₃

**Figure 3.** Gantt chart of the initial scheduling.**Figure 4.** Gantt chart of the initial scheduling.

al., 1992; Aarts et al., 1994) including some heuristic and meta-heuristic algorithms.

The experimental results compared with other 14 algorithms are listed in Table 3a and 3b. As can be seen in Table 3a and 3b, size means the name of each test

problem, BKS represents the best known solution for the instance, best denotes the best solution found by the algorithm, and RD represents the percentage of the deviation with respect to the best known solution for DABC, MPSO, and HIA algorithm. From the table, we can see that the proposed algorithm is able to find the best known solution in 38 instances, that is, 88% of the instances, and the deviation of the minimum found makespan from the best known solution is only on average 0.08%. DABC presents an improvement with respect to almost all other algorithms. These better solutions show that the DABC is very suitable for solving the JSSP. Table 4 shows the number of instances solved (NIS) and the average deviation of the minimum found makespan with respect to the BKS. The ARD was calculated for the DABC algorithm, and for the other algorithm (OA). The last column (Improvement) denotes the reduction in ARD obtained by the DABC with respect to each of other algorithm. From the table, we can find that the DABC can obtain the ARD of 0.08% for solving 43 instances. DABC can perform much better than almost all other algorithms, except for the approach proposed by Nowicki and Smutnicki.

We use the same parameters to test the DABC for instances FT06, FT10, FT20 and the first instance of every type instance set as test benchmarks. Each instance is executed 15 times for DABC algorithm. The results are listed in Table 5. Mean denotes the mean solution. MRD means the relative deviation of the mean solution. SD represents the standard deviation of the solutions. Best denotes the best solution. BRD means the relative deviation of the best solution. For the small scale instance, The MRD is zero and then MRD is no more than 0.6% for most large scale instance.

To illustrate the experimental results more intuitively, we set the FT06-FT20 as examples. FT06 is easy to be solved. The FT10 and FT 20 have a lot of local optimal so that they are difficult to find the optimal solution. The

Table 3a. Comparisons of result between the DABC with other algorithms.

Instance	Size(n*m)	BKS	This paper		Ge hong wei et al		Tsung-Lieh et al		Ombuki	Coello	Aiex	Wang
			BEST	RD	HIA BEST	HIA RD	MPSO BEST	MPSO RD	LSGA	AIS	GP+PR	MGA
FT06	6*6	55	55	0.00	55	0.00	55	0.00	-	-	55	55
FT10	10*10	930	930	0.00	930	0.00	930	0.00	-	941	930	930
FT20	20*5	1165	1165	0.00	1165	0.00	1165	0.00	-	-	1165	1165
LA01	10*5	666	666	0.00	666	0.00	666	0.00	-	666	666	666
LA02	10*5	655	655	0.00	655	0.00	655	0.00	-	655	655	-
LA03	10*5	597	597	0.00	597	0.00	597	0.00	-	597	597	-
LA04	10*5	590	590	0.00	590	0.00	590	0.00	-	590	590	-
LA05	10*5	593	593	0.00	593	0.00	593	0.00	-	593	593	-
LA06	15*5	926	926	0.00	926	0.00	926	0.00	-	926	926	926
LA07	15*5	890	890	0.00	890	0.00	890	0.00	-	890	890	-
LA08	15*5	863	863	0.00	863	0.00	863	0.00	-	863	863	-
LA09	15*5	951	951	0.00	951	0.00	951	0.00	-	951	951	-
LA10	15*5	958	958	0.00	958	0.00	958	0.00	-	958	958	-
LA11	20*5	1222	1222	0.00	1222	0.00	1222	0.00	-	-	1222	1222
LA12	20*5	1039	1039	0.00	1039	0.00	1039	0.00	-	-	1039	-
LA13	20*5	1150	1150	0.00	1150	0.00	1150	0.00	-	-	1150	-
LA14	20*5	1292	1292	0.00	1292	0.00	1292	0.00	-	-	1292	-
LA15	20*5	1207	1207	0.00	1207	0.00	1207	0.00	-	-	1207	-
LA16	10*10	945	945	0.00	945	0.00	945	0.00	959	945	945	945
LA17	10*10	784	784	0.00	784	0.00	784	0.00	792	785	784	-
LA18	10*10	848	848	0.00	848	0.00	848	0.00	857	848	848	-
LA19	10*10	842	842	0.00	842	0.00	842	0.00	860	848	842	-
LA20	10*10	902	902	0.00	902	0.00	902	0.00	907	907	902	-
LA21	15*10	1046	1046	0.00	1046	0.00	1046	0.00	1114	-	1057	1058
LA22	15*10	927	927	0.00	932	0.54	932	0.54	989	-	927	-
LA23	15*10	1032	1032	0.00	1032	0.00	1032	0.00	1035	-	1032	-
LA24	15*10	935	938	0.32	950	1.66	941	0.64	1032	-	954	-
LA25	15*10	977	977	0.00	979	0.20	977	0.00	1047	1022	984	-
LA26	20*10	1218	1218	0.00	1218	0.00	1218	0.00	1307	-	1218	1218
LA27	20*10	1235	1235	0.00	1256	1.70	1239	0.32	1350	-	1269	-
LA28	20*10	1216	1216	0.00	1227	0.90	1216	0.00	1312	1277	1225	-
LA29	20*10	1152	1167	1.30	1184	2.78	1173	1.82	1311	1248	1203	-
LA30	20*10	1355	1355	0.00	1355	0.00	1355	0.00	1451	-	1355	-
LA31	30*10	1784	1784	0.00	1784	0.00	1784	0.00	1784	-	1784	1784
LA32	30*10	1850	1850	0.00	1850	0.00	1850	0.00	1850	-	1850	-
LA33	30*10	1719	1719	0.00	1719	0.00	1719	0.00	1745	-	1719	-
LA34	30*10	1721	1721	0.00	1721	0.00	1721	0.00	1784	-	1721	-
LA35	30*10	1888	1888	0.00	1888	0.00	1888	0.00	1958	1903	1888	-
LA36	15*15	1268	1268	0.00	1281	1.03	1278	0.79	1358	1323	1287	1291
LA37	15*15	1397	1407	0.72	1415	1.72	1411	1.00	1517	-	1410	-
LA38	15*15	1196	1202	0.50	1215	1.42	1208	1.00	1362	1274	1218	-
LA39	15*15	1233	1233	0.00	1246	1.05	1233	0.00	1391	1270	1248	-
LA40	15*15	1222	1228	0.49	1240	1.47	1225	0.25	1323	1258	1244	-
FT06	6*6	55	55	0.00	55	-	55	55	55	-	-	-
FT10	10*10	930	930	0.00	938	1016	966	930	930	960	-	-
FT20	20*5	1165	1165	0.00	1169	-	1178	1165	1165	1249	-	-

Table 3a. Comparisons of result between the DABC with other algorithms.

Instance	Size (n*m)	BKS	This paper		Binato	Sabuncuoglu	Dauzere	Nuijten	Nowocki	Dorndorf and Pesh		
			BEST	RD	GRASP	Beam search	Tabu search	RCS	Tabu search	PGA	SBGA(40)	SBGA(60)
LA01	10*5	666	666	0.00	666	666	666	666	666	666	666	-
LA02	10*5	655	655	0.00	655	704	655	655	655	681	666	-
LA03	10*5	597	597	0.00	604	650	603	597	597	620	604	-
LA04	10*5	590	590	0.00	590	620	590	590	590	620	590	-
LA05	10*5	593	593	0.00	593	593	593	593	593	593	593	-
LA06	15*5	926	926	0.00	926	926	926	926	926	926	926	-
LA07	15*5	890	890	0.00	890	890	890	890	890	890	890	-
LA08	15*5	863	863	0.00	863	863	863	863	863	863	863	-
LA09	15*5	951	951	0.00	951	951	951	951	951	951	951	-
LA10	15*5	958	958	0.00	958	958	958	958	958	958	958	-
LA11	20*5	1222	1222	0.00	1222	1222	1222	1222	1222	1222	1222	-
LA12	20*5	1039	1039	0.00	1039	1039	1039	1039	1039	1039	1039	-
LA13	20*5	1150	1150	0.00	1150	1150	1150	1150	1150	1150	1150	-
LA14	20*5	1292	1292	0.00	1292	1292	1292	1292	1292	1292	1292	-
LA15	20*5	1207	1207	0.00	1207	1207	1207	1207	1207	1237	1207	-
LA16	10*10	945	945	0.00	946	988	945	945	945	1008	961	961
LA17	10*10	784	784	0.00	784	827	784	784	784	809	787	784
LA18	10*10	848	848	0.00	848	881	848	848	848	916	848	848
LA19	10*10	842	842	0.00	842	882	842	848	842	880	863	848
LA20	10*10	902	902	0.00	907	948	902	907	902	928	911	910
LA21	15*10	1046	1046	0.00	1091	1154	1057	1069	1047	1139	1074	1074
LA22	15*10	927	927	0.00	960	985	948	937	927	998	935	936
LA23	15*10	1032	1032	0.00	1032	1051	1032	1032	1032	1072	1032	1032
LA24	15*10	935	938	0.32	978	992	946	942	939	1014	960	957
LA25	15*10	977	977	0.00	1028	1073	992	981	977	1014	1008	1007
LA26	20*10	1218	1218	0.00	1271	1269	1218	1218	1218	1278	1219	1218
LA27	20*10	1235	1235	0.00	1320	1316	1269	1285	1236	1378	1272	1269
LA28	20*10	1216	1216	0.00	1293	1373	1237	1216	1216	1327	1240	1241
LA29	20*10	1152	1167	1.30	1293	1252	1215	1208	1160	1336	1204	1210
LA30	20*10	1355	1355	0.00	1368	1435	1355	1355	1355	1411	1355	1355
LA31	30*10	1784	1784	0.00	1784	1784	1784	1784	1784	-	-	-
LA32	30*10	1850	1850	0.00	1850	1850	1850	1850	1850	-	-	-
LA33	30*10	1719	1719	0.00	1719	1719	1719	1719	1719	-	-	-
LA34	30*10	1721	1721	0.00	1753	1780	1721	1721	1721	-	-	-
LA35	30*10	1888	1888	0.00	1888	1888	1888	1888	1888	-	-	-
LA36	15*15	1268	1268	0.00	1334	1401	1313	1292	1268	1373	1317	1317
LA37	15*15	1397	1407	0.72	1457	1503	1456	1411	1407	1498	1484	1446
LA38	15*15	1196	1202	0.50	1267	1297	1254	1278	1196	1296	1251	1241
LA39	15*15	1233	1233	0.00	1290	1369	1245	1233	1233	1351	1282	1277
LA40	15*15	1222	1228	0.49	1259	1347	1242	1247	1229	1321	1274	1252

Table 3b. Comparisons of result between the DABC with other algorithms.

Instance	Size(n*m)	BKS	This paper		Croce	Goncalves et al. (2005)			Stoter et.al	Aarts eal	
			BEST	RD	GA	Param Active	HGA Nondelay	Active	1992	GLS1	GLS2
FT06	6*6	55	55	0.00	-	55	55	55	-	-	-
FT10	10*10	930	930	0.00	946	930	951	945	952	935	945
FT20	20*5	1165	1165	0.00	1178	1165	1178	1173	-	1165	1167
LA01	10*5	666	666	0.00	666	666	666	666	666	666	666
LA02	10*5	655	655	0.00	666	655	655	655	-	668	659
LA03	10*5	597	597	0.00	666	597	604	603	-	613	609
LA04	10*5	590	590	0.00	-	590	590	598	-	599	594
LA05	10*5	593	593	0.00	-	593	593	593	-	593	593
LA06	15*5	926	926	0.00	926	926	926	926	-	926	923
LA07	15*5	890	890	0.00	-	890	890	890	-	890	890
LA08	15*5	863	863	0.00	-	863	863	863	-	863	863
LA09	15*5	951	951	0.00	-	951	951	951	-	951	951
LA10	15*5	958	958	0.00	-	958	958	958	-	958	958
LA11	20*5	1222	1222	0.00	1222	1222	1222	1222	-	1222	1222
LA12	20*5	1039	1039	0.00	-	1039	1039	1039	-	1039	1039
LA13	20*5	1150	1150	0.00	-	1150	1150	1150	-	1150	1150
LA14	20*5	1292	1292	0.00	-	1292	1292	1292	-	1292	1292
LA15	20*5	1207	1207	0.00	-	1207	1207	1207	-	1207	1207
LA16	10*10	945	945	0.00	979	945	973	947	981	977	977
LA17	10*10	784	784	0.00	-	784	792	784	794	791	791
LA18	10*10	848	848	0.00	-	848	855	848	860	856	858
LA19	10*10	842	842	0.00	-	842	851	852	860	863	859
LA20	10*10	902	902	0.00	-	907	926	912	-	913	916
LA21	15*10	1046	1046	0.00	1097	1046	1079	1074	-	1084	1085
LA22	15*10	927	927	0.00	-	935	950	962	-	954	944
LA23	15*10	1032	1032	0.00	-	1032	1032	1032	-	1032	1032
LA24	15*10	935	938	0.32	-	953	970	955	-	970	981
LA25	15*10	977	977	0.00	-	986	1013	1014	-	1016	1010
LA26	20*10	1218	1218	0.00	1231	1218	1218	1237	-	1240	1236
LA27	20*10	1235	1235	0.00	-	1256	1282	1280	-	1308	1300
LA28	20*10	1216	1216	0.00	-	1232	1250	1250	-	1281	1265
LA29	20*10	1152	1167	1.30	-	1196	1206	1226	-	1290	1260
LA30	20*10	1355	1355	0.00	-	1355	1355	1355	-	1402	1386
LA31	30*10	1784	1784	0.00	1784	1784	1784	1784	-	1784	1784
LA32	30*10	1850	1850	0.00	-	1850	1850	1850	-	1850	1850
LA33	30*10	1719	1719	0.00	-	1719	1719	1719	-	1719	1719
LA34	30*10	1721	1721	0.00	-	1721	1721	1721	-	1737	1730
LA35	30*10	1888	1888	0.00	-	1888	1888	1888	-	1894	1890
LA36	15*15	1268	1268	0.00	1305	1279	1303	1313	1305	1324	1311
LA37	15*15	1397	1407	0.72	-	1408	1437	1444	1458	1449	1450
LA38	15*15	1196	1202	0.50	-	1219	1252	1228	1239	1285	1283
LA39	15*15	1233	1233	0.00	-	1246	1250	1265	1258	1279	1279
LA40	15*15	1222	1228	0.49	-	1241	1252	1246	1258	1273	1260

convergence rate of DABC algorithm could be found in Figures 5- 7 for three instances. Figures 8-10 show the Gantt chart of the best solution for three instances.

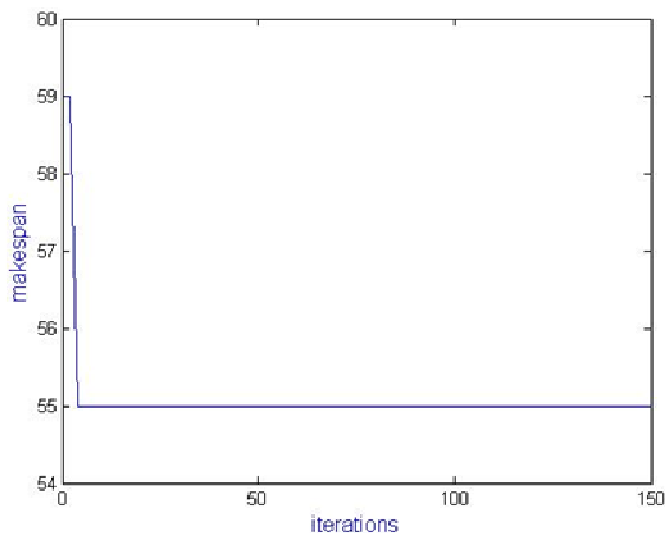
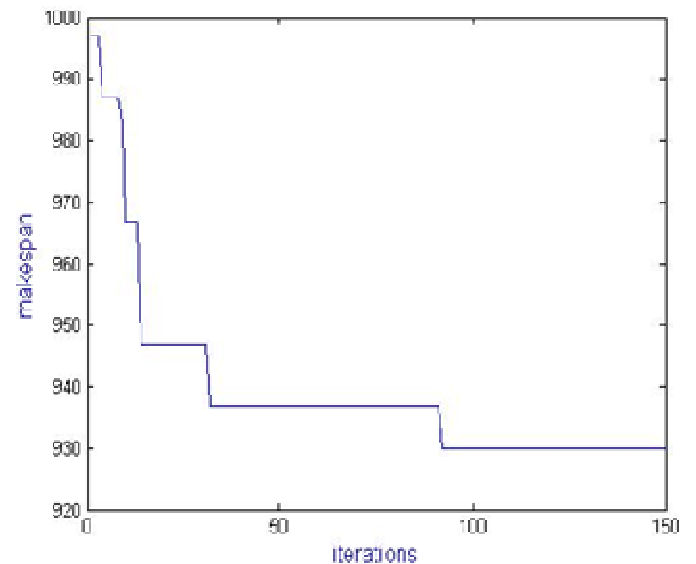
We can define the utilization percentage of the i^{th} machine for JSSP as follows: Where IT_i denotes the total idle time when the last operation is processed, and

Table 4. Average relative deviation to the BKS.

Algorithm	NIS	ARD	Improvement	Algorithm
		OA (%)	DABC (%)	DABC (%)
Problem and heuristic space				
Storer et al. (1992)	11	2.44	0.15	2.29
Genetic algorithm				
Aarts et al. (1994)-GLS1	42	1.98	0.08	1.9
Aarts et al. (1994)-GLS2	42	1.71	0.08	1.63
Croce et al. (1995)	12	2.37	0	2.37
Dorndorf and Pesch (1995)-PGA	37	4.62	0.09	4.53
Dorndorf and Pesch (1995)-SBGA(40)	35	1.43	0.10	1.33
Dorndorf and Pesch (1995)-SBGA(60)	20	1.96	0.17	1.79
Ombuki and Ventresca (2004)-LSGA	25	5.69	0.13	5.56
Goncalves et al. (2005)-Param Active	43	0.40	0.08	1.32
Goncalves et al. (2005)-HGA-non delay	43	1.20	0.08	1.12
Goncalves et al. (2005)-Active	43	1.12	0.08	1.04
GRASP				
Binato et al. (2002)	43	1.77	0.08	1.69
Aiex et al. (2001)	43	0.44	0.08	1.36
Hybrid genetic and simulated annealing				
Wang and zheng (2001)	11	0.28	0	0.28
Hybrid particle swarm optimization and AIS				
Ge Hong wei et al. (2008)	43	0.33	0.08	1.25
AIS				
Coello et al. (2003)-AIS	24	1.59	0.10	1.49
Particle swarm optimization				
Tsung-Lieh et al. (2009)-MPSO	43	0.15	0.08	0.07
Tabu search				
Nowicki and Smutniki (1996)	43	0.06	0.08	-0.02
Dauzere (1997)	43	0.87	0.08	0.79
Beam search				
Sabuncuoglu and Bayiz (1999)	41	4.35	0.08	4.27
Constraint satisfaction				
Nuijten and Aarts (1996)	41	0.61	0.08	0.53

Table 5. Average experimental result using the DABC.

FSSP	Size (Js*Ms)	BKS	Mean	MRD(%)	SD	BEST	BRD
FT06	6*6	55	55	0.00	0.00	55	0.00
FT10	10*10	930	930.7	0.08	2.12	930	0.00
FT20	20*5	1165	1168.9	0.33	6.27	1165	0.00
LA01	10*5	666	666	0.00	0.00	666	0.00
LA06	15*5	926	926	0.00	0.00	926	0.00
LA11	20*5	1222	1222	0.00	0.00	1222	0.00
LA16	10*10	945	945	0.00	0.00	945	0.00
LA21	15*10	1046	1050.9	0.47	3.38	1046	0.00
LA26	20*10	1218	1218	0.00	0.00	1218	0.00
LA31	30*10	1784	1784	0.00	0.00	1784	0.00
LA36	15*15	1268	1274.5	0.51	5.50	1268	0.00

**Figure 5.** Convergence curves for DABC for FT06.**Figure 6.** Convergence curves for DABC for FT10.

PT_i represents the total processing time of machine i . Figures 11-13 show the utilization percentage of each machine for three instances.

From these three pictures, we can find that there are many idle time for some machine between the finish time of the last operation and the makespan. Therefore, let DT_i present the idle time between the finish time of the last operation and the makespan.

$$DT_i + PT_i + IT_i = makespan \quad \forall i \in \{1, 2, \dots, M\}$$

Define the percentage of the total processing, idle, dormancy time for the entire machines as follows: Figures 14 –16 show the pie chart of percentage of different times.

Conclusion

A promising hybrid algorithm that combines the DABC and fast local search, the pairwise based local search is proposed to solve JSSP with minimization of the makespan. Firstly, we present a food source as a discrete job permutation and apply the discrete operation to generate new neighborhood food source for the three different bees. Secondly, three mutation operations are proposed to apply the DABC to solve the job shop scheduling. Thirdly, fast local search is proposed to enhance the individual of the DE with a certain probability. Fourthly, the pairwise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum. Experimental

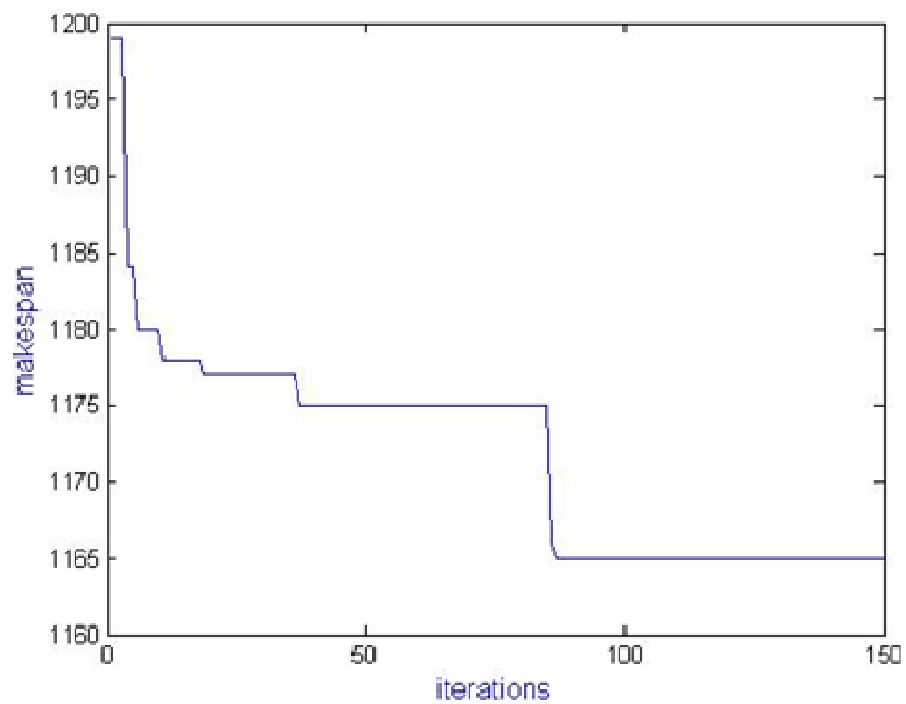


Figure 7. Convergence curves for DABC for FT20.

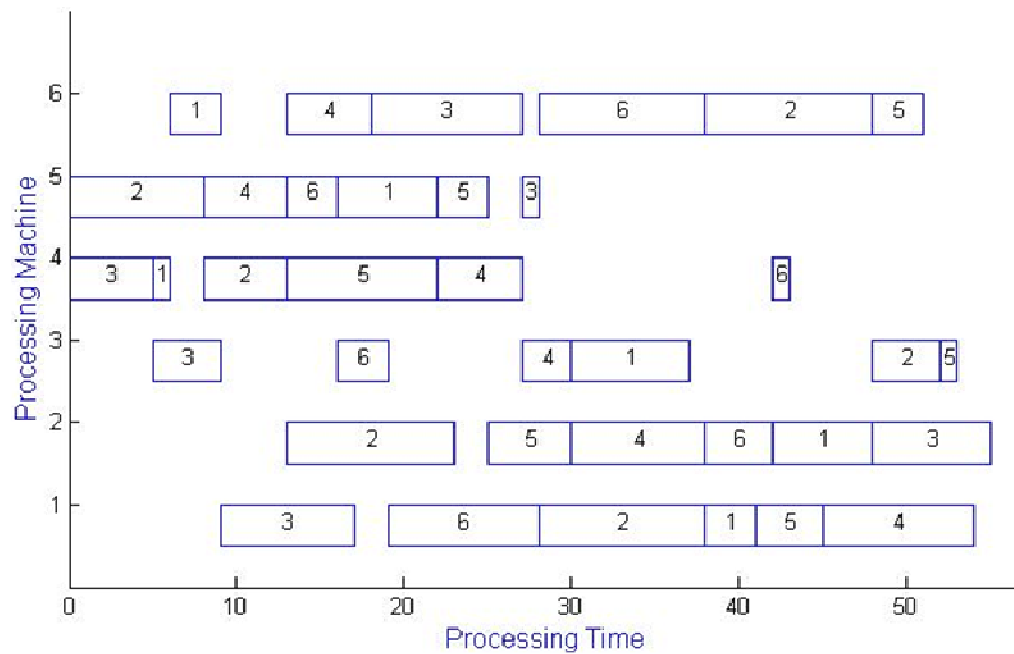


Figure 8. Gantt chart of an optimal schedule for FT06.

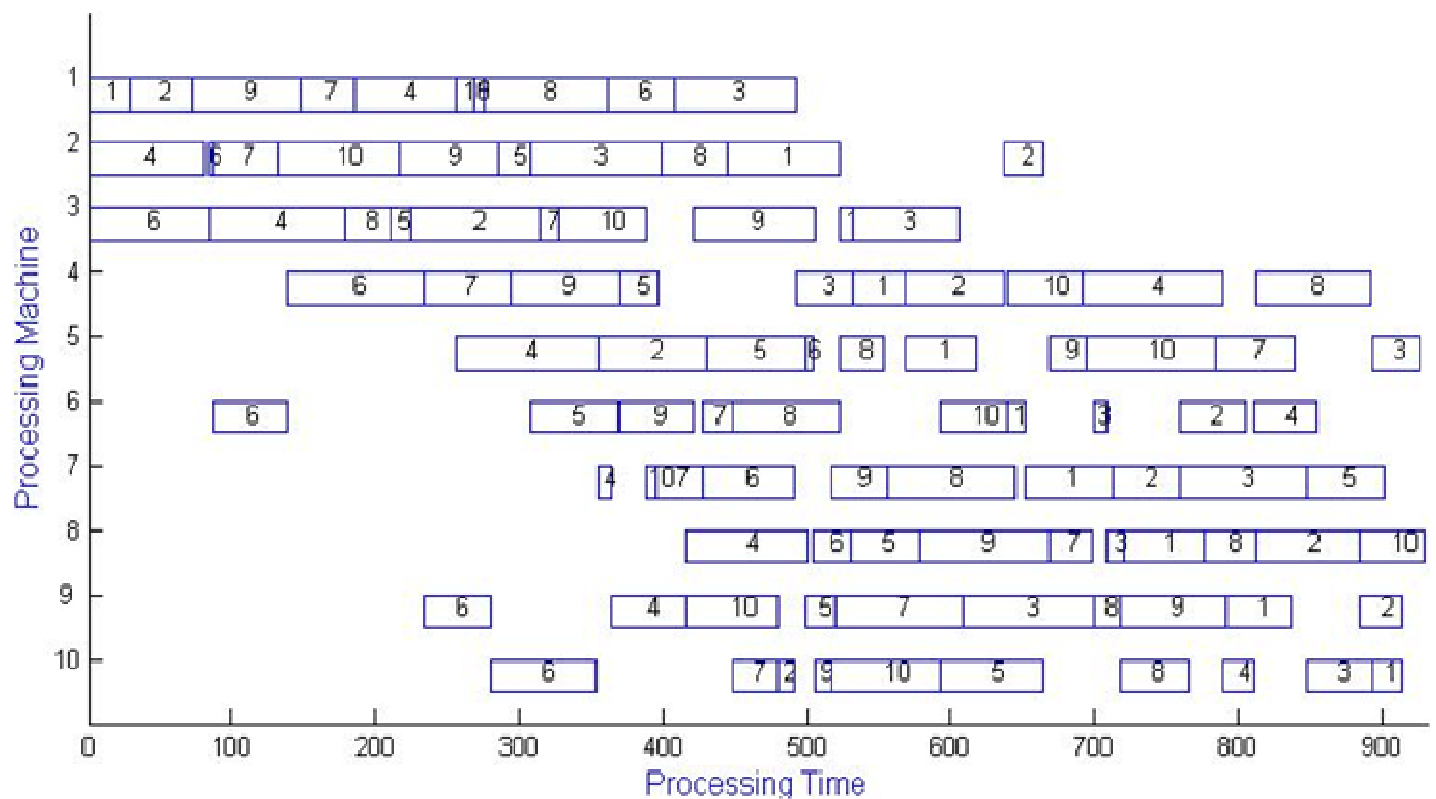


Figure 9. Gantt chart of an optimal schedule for FT10.

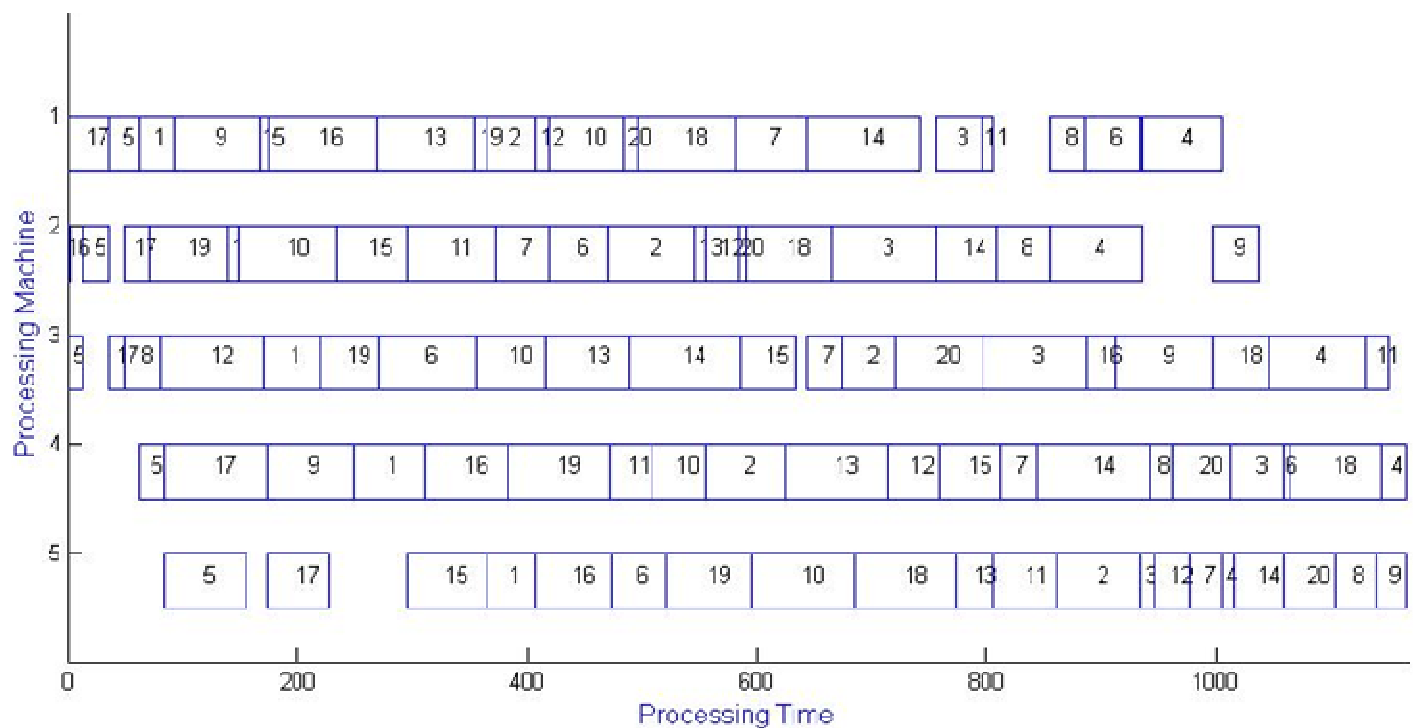


Figure 10. Gantt chart of an optimal schedule for FT20.

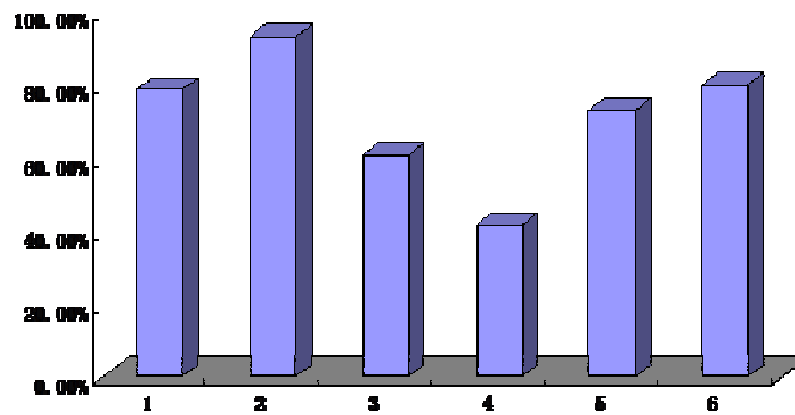


Figure 11. Column chart of percentage of utilization for FT06.

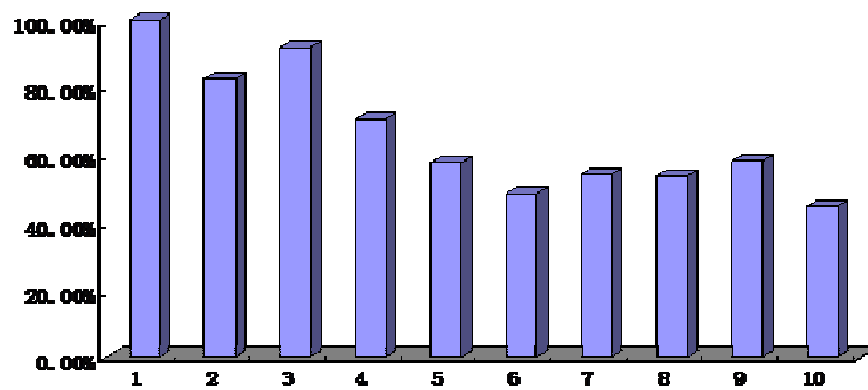


Figure 12. Column chart of percentage of utilization for FT10.

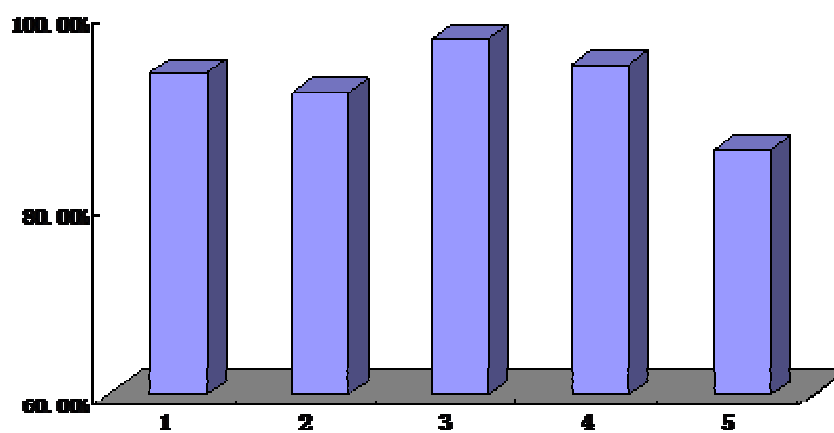


Figure 13. Column chart of percentage of utilization for FT20.

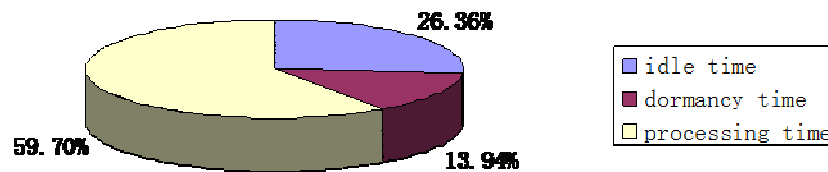


Figure 14. Column chart of percentage of utilization for FT06.

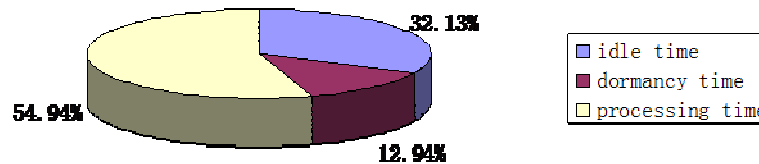


Figure 15. Column chart of percentage of utilization for FT10.

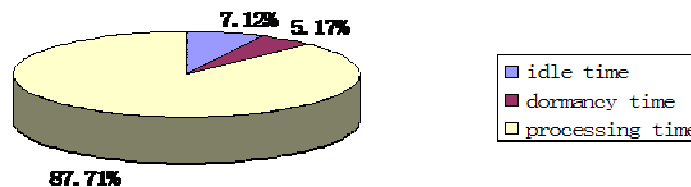


Figure 16. Column chart of percentage of utilization for FT20.

results and comparisons show the effectiveness of the proposed DABC for JSSP. Moreover, the further work is to study the theoretical aspects as well as the performance of the technique. The other problem is to extend the algorithm to solve other combination problem such as open shop scheduling.

ACKNOWLEDGMENTS

This research is fully supported by the National Natural Science Foundation of China under Grant Nos. 60803102, and also funded by NSFC Major Research Program 60496321: Basic Theory and Core Techniques of Non Canonical Knowledge.

REFERENCES

- Aarts EHL, Laarhoven Van PJM, Lenstra JK, Ulder NLJ (1994). A computational study of local search algorithms for job shop scheduling. *ORSA J. Comput.*, 6: 118-125.
- Aiex RM, Binato S, Resende MGC (2003). Parallel GRASP with path-relinking for job shop scheduling. *Parallel Comput.*, 29: 393-430.
- Aldowaisan T, Allahverdi A (2003). New heuristics for no-wait flowshops to minimize makespan. *Comput. Oper. Res.*, 30: 1219-1231.
- Amirthagadeswaran KS, Arunachalam VP (2006). Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. *Int. J. Adv. Manuf. Technol.*, 28: 532-540.
- Beasley JE (1990). OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41: 1069-1072.
- Binato S, Hery WJ, Loewenstern DM, Resende MGC (2001). A GRASP for job shop scheduling, in *Essays and Surveys in Metaheuristics*. Boston, MA: Kluwer, pp. 59-80.
- Coello CAC, Rivera DC, Cortes NC (2003). Use of an artificial immune system for job shop scheduling. in *Proc. 2nd Int. Conf. Artif. Immune Syst.*, 2787: 1-10.
- Croce FD, Tadei R, Volta G (1995). A genetic algorithm for the job shop problem. *Comput. Oper. Res.*, 22: 15-24.
- Dauzere PS, Pauli J (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Ann. Oper. Res.*, 70: 281-306.
- David HW, William GM (1997). No Free Lunch Theorems for Optimization, *IEEE trans. Evolut. Comput.*, 1: 67-82.
- Dorndorf U, Pesch E (1995). Evolution based learning in a job shop scheduling environment. *Comput. Oper. Res.*, 22: 25-40.
- Gang X, Wu ZM (2004). Deadlock-free scheduling strategy for automated production cell. *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, 34(1): 113-122.
- Garey MR, Johnson DS, Sethi R (1976). The complexity of flow shop and job shop scheduling. *Math. Oper. Res.*, 1: 117-129.
- Ge HW, Sun L, Liang YC, Qian F (2008). An Effective PSO and AIS-Based Hybrid Intelligent Algorithm for Job-Shop Scheduling. *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, 38: 358-368.
- Geyik F, Cedimoglu IH (2004). The strategies and parameters of tabu search for job-shop scheduling. *J. Intell. Manuf.*, 15: 439-448.
- Goncalves JF, Mendes JJDM, Resende MGC (2005). A hybrid genetic algorithm for the job shop scheduling problem. *Eur. J. Oper. Res.*, 167: 77-95.
- Hajri S, Liouane N, Hammadi S, Borne P (2000). A controlled genetic algorithm by fuzzy logic and belief functions for job shop scheduling. *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 30(5): 812-818.

- Karaboga D (2005). An idea based on honey bee swarm for numerical optimization, technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.
- Karaboga D, Akay B (2009). A Comparative Study of Artificial Bee Colony Algorithm, *Appl. Math. Comput.*, 214: 108-132.
- Karaboga D, Basturk B (2007). A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *J. Glo. Opti.*, 39: 459-171.
- Karaboga D, Basturk B (2008). On The Performance of Artificial Bee Colony (ABC) Algorithm. *Appl. Soft. Com.*, 8: 687-697.
- Karaboga D, Ozturk C (2010). Fuzzy clustering with artificial bee colony algorithm. *Sci. Res. Essays*, 5(14): 1899-1902.
- Kolonko M (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *Eur. J. Oper. Res.*, 113: 123-136.
- Lian ZG, Jiao B, Gu XS (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Appl. Math. Comput.*, 183: 1008-1017.
- Lin TL, Horng SJ, Kao TW, Chen YH, Run RS, Chen RJ, Lai JL, Kuo IH (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Exp. Syst. Appl.*, 37: 2629-2636.
- Liu TK, Tsai JT, Chou JH (2006). Improved genetic algorithm for the job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.*, 27: 1021-1029.
- Low C, Yeh JY, Huang KI (2004). A robust simulated annealing heuristic for flow shop scheduling problems. *Int. J. Adv. Manuf. Technol.*, 23: 762-767.
- Nowicki E, Smutnicki C (1996). A fast taboo search algorithm for the job shop problem. *Manage. Sci.*, 42: 797-813.
- Nuijten WPW, Aarts EHL (1996). Computational study of constraint satisfaction for multiple capacitated job shop scheduling. *Eur. J. Oper. Res.*, 90: 269-284.
- Ombuki BM, Ventresca M (2004). Local search genetic algorithms for the job shop scheduling problem. *Appl. Intell.*, 21: 99-109.
- Pan QK, Tasgetiren MF, Suganthan PN, Chua TJ (2010). A Discrete Artificial Bee Colony Algorithm for the Lot-streaming Flow Shop Scheduling Problem. *Infor. Sci.* doi:10.1016/j.ins.2009.12.025.
- Ponnambalam SG, Aravindan P, Rajesh SV (2000). A tabu search algorithm for job shop scheduling. *Int. J. Adv. Manuf. Technol.*, 16: 765-771.
- Sabuncuoglu I, Bayiz M (1999). Job shop scheduling with beam search. *Eur. J. Oper. Res.*, 118: 390-412.
- Stadtler H (2005). Supply chain management and advanced planning basics, overview and challenges. *Eur. J. Oper. Res.*, 163: 575-588.
- Storer RH, Wu SD, Park I (1992). Genetic Algorithms in Problem Space for Sequencing Problems, *Pro. a Joint US-German Conf. on Oper. Res. Prod. Plan. Con.*, pp. 584-597.
- Suresh RK, Mohanasundaram KM (2006). Pareto archived simulated annealing for job shop scheduling with multiple objectives. *Int. J. Adv. Manuf. Technol.*, 29: 184-196.
- Wang L, Zheng DZ (2002). A modified genetic algorithm for job shop scheduling. *Int. J. Adv. Manuf. Technol.*, 20: 72-76.
- Wang TY, Wu KB (2000). A revised simulated annealing algorithm for obtaining the minimum total tardiness in job shop scheduling problems. *Int. J. Syst. Sci.*, 31: 537-542.
- Watson JP, Beck JC, Howe AE (2003). Problem difficulty for tabu search in job-shop scheduling. *Artif. Intell.*, 143: 189-217.
- Xia WJ, Wu ZM (2006). A hybrid particle swarm optimization approach for the job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.*, 29: 360-366.
- Yang S, Wang D (2000). Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling. *IEEE Trans. Neural Netw.* 11: 474-486.
- Yang S, Wang D (2001). A new adaptive neural network and heuristics hybrid approach for job-shop scheduling. *Comput. Oper. Res.*, 28: 955-971.
- Yin MH, Zou TT, Gu WX (2010). Reverse bridge theorem under constraint partition. *Math. Pro. Eng.*, Article ID 617398, doi:10.1155/2010/617398.
- Yu HB, Liang W (2001). Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. *Comput. Ind. Eng.*, 39: 337-356.
- Zhang J, Hu XM, Tan X, Zhong JH, Huang Q (2006). Implementation of an ant colony optimization technique for job shop scheduling problem. *Trans. Inst. Meas. Control.*, 28: 93-108.