J Intell Manuf (2018) 29:451–462
https://doi.org/10.1007/s10845-015-1121-x

Parallel bat algorithm for optimizing makespan in job shop scheduling problems

Thi-Kien Dao[1] · Tien-Szu Pan[1] · Trong-The Nguyen[1] · Jeng-Shyang Pan[2]

Received: 14 March 2015 / Accepted: 29 June 2015 / Published online: 14 July 2015

Abstract Parallel processing plays an important role in efficient and effective computations of function optimization. In this paper, an optimization algorithm based on parallel versions of the bat algorithm (BA), random-key encoding scheme, communication strategy scheme and makespan scheme is proposed to solve the NP-hard job shop scheduling problem. The aim of the parallel BA with communication strategies is to correlate individuals in swarms and to share the computation load over few processors. Based on the original structure of the BA, the bat populations are split into several independent groups. In addition, the communication strategy provides the diversity-enhanced bats to speed up solutions. In the experiment, forty three instances of the benchmark in job shop scheduling data set with various sizes are used to test the behavior of the convergence, and accuracy of the proposed method. The results compared with the other methods in the literature show that the proposed scheme increases more the convergence and the accuracy than BA and particle swarm optimization.

Keywords Parallel bat algorithm · Job shop scheduling problem · Swarm intelligence

Introduction

The idea of the parallel bat algorithm (BA) is to divide the artificial bats into independent subpopulations so that they can share the computation load. The parallel processing also had been applied for the existing methods, including in the ant colony system with communication strategies (Chu et al. 2004), a parallel particle swarm optimization algorithm with communication strategies (Chu et al. 2005), parallel cat swarm optimization (Tsai et al. 2008), the island-model genetic algorithm (Whitley et al. 1998), and parallel genetic algorithm (Abramson and Abela 1991). The global search capacity could be extended and the accuracy could be increased from the parallelized subpopulation structure of artificial agents more than that the original structure. The parallelized strategies simply share the computation load over several processors. The sum of the computation time for all processors can be reduced compared with the single processor works on the same optimum problem (Kuck 1977). Besides, the no free lunch theorem (Wolpert and Macready 1997) has logically proven that there is no meta-heuristic algorithm best suitable for solving all optimization problems. In other words, a particular meta-heuristic algorithm may show highly promising results on a set of problems, but the same algorithm may show poor performance on a different set of problems (Wolpert and Macready 2005). This theorem makes the field of meta-heuristic algorithm and the study of its applications highly active. This also motivates us to consider the strength of the algorithm in parallel processing for solving complicated and difficult scheduling problems (Garey and Johnson 1990).

Scheduling can be defined as a problem of finding an optimal sequence to execute a finite set of operations of satisfied constraints (Błażewicz et al. 1996). It is a decision-making process of allocating resources over time to perform a collection of required tasks. Effective scheduling plays a vital role in the growth of any kind of industries. Different types of scheduling problems were addressed in the literature with unlike performance measures (Behnamian and Fatemi Ghomi 2014; Gen and Lin 2014). Single machine

✉ Jeng-Shyang Pan
jengshyangpan@fjut.edu.cn

[1] Department of Electronics Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, ROC

[2] College of Information Science and Engineering, Fujian University of Technology, Fuzhou, China

scheduling (Çakar 2011), flow shop scheduling (Mirabi et al. 2013), parallel machine scheduling (Ying et al. 2012), job shop scheduling (Meeran and Morshed 2012) are some of the important types of scheduling environments. Among them, the job shop scheduling problem (JSSP) is considered in this paper. Minimization of makespan, total of flow time, mean of flow time, total earliness, total tardiness, weighted earliness, and tardiness and number of tardy jobs are the important objective functions of scheduling problems. The objective of this paper is to minimize makespan. In recent years, an interest in using meta-heuristic methods to solve JSSP has been growing rapidly, such as simulated annealing (SA; Laarhoven et al. 1992; Song et al. 2012), tabu search (TS; Dell'Amico and Trubian 1993; Geyik and Cedimoglu 2004; Zhang et al. 2007), genetic algorithms (GA; Davis 1985; Moin et al. 2015; Gonçalves et al. 2005; Cheng et al. 1996), particle swarm optimization (PSO; Lian et al. 2006; Lin et al. 2010), artificial immune system and their hybrids (AIS; Coello et al. 2003; Qiu and Lau 2014; Luh and Chueh 2009; Zhang and Wu 2010), discrete artificial bee colony (DABC; Yin et al. 2011), discrete imperialist competitive algorithm (Hosseini and Al Khaled 2014) and hybrid imperialist competitive algorithm(Hosseini et al. 2014). These approaches comprise the emergence of promise for conquering the combinatorial exploration in a variety of decision-making arenas.

A new meta-heuristic optimization algorithm, namely, BA and its parallel version based on swarm intelligence and the inspiration for observing the searching for the prey of the bats was introduced as in (Yang 2010; Tsai et al. 2014). The key advantage of the BA is that it can provide very quick convergence at a very initial stage by switching from exploration to exploitation (Yang and He 2013). It is potentially more powerful than PSO and GA as well as harmony search. The primary reason is that BA uses a good combination of major advantages of these algorithms in some way. Moreover, PSO and harmony search are the special cases of the BA under appropriate simplifications.

In this paper, the concept of a parallel processing is applied to BA and a communication strategy for PBA is proposed to solve JSSP. A set of independent jobs must be processed on a set of available machines in JSSP. Each job is a sequence of operations. Each operation requires a predefined machine. These are all handled efficiently with PBA.

The rest of this paper is organized as follows. The problem of job shop scheduling is reviewed in "The job shop scheduling problem" section. The analysis and designs for PBA with the communication strategy and its experimental results are presented in "Parallelized bat algorithm with a communication strategy" section. The application of PBA to the JSSP problems is presented in "Parallel bat algorithm for JSSP" section. "Experimental results" section shows further results of testing JSSP. Finally, the conclusion is summarized in "Conclusion" section.

The job shop scheduling problem

The JSSP was defined (Cheng et al. 1996; Gonçalves et al. 2005) as follows. There are a machine set $\mathrm{M} = \{\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_\mathrm{m}\}$ and a job set $\mathrm{J} = \{\mathrm{J}_1, \mathrm{J}_2, \ldots, \mathrm{J}_\mathrm{n}\}$. Each job, J_k, must go through m machines to complete its work. One job consists of a set of operations, and the operation order for the machines is predetermined. Each operation uses one of the m machines to complete one job's work for a fixed time interval. Once an operation is processed on a given machine, it cannot be interrupted before it finishes the procedure. The sequence of the operations of a job should be predefined and maybe different for any job. Every job has a sequence of m operations. Each machine can process only one operation during the time interval. A schedule is the assignment of operations to time slots on a machine. The objective of the JSSP is to find an appropriate schedule. A good schedule is an appropriate operation permutation for all jobs that can minimize the makespan or one that minimizes the idle time of machines. The makespan is denoted as C_{max}. It is the maximum total completion time of the final operation in the schedule of $n \times m$ operations. A set of the operations is denoted as $\mathrm{O} = \{0, 1, 2, \ldots, n \times m + 1\}$. The operations 0 and $n \times m + 1$ are the dummy operations which represent the initial and the last operations, respectively. A dummy operation is used to model the JSSP problem and does not need any processing time. Furthermore, let T and P denote the fixed processing time and the preceding operations.

For an $n \times m$ JSSP, the problem can be modeled on a set of m machines to process a set of $n \times m$ operations. An operation can be scheduled for an appointed free machine. A precedence constraint is used to schedule Operation *i-th* after finishing P_i. t_i is the processing time of Operation i on a given machine.

The objective fitness function can be formulated to minimize makespan C_{max} as follows:

$$Fitness = minimize\ O_{n\times m+1}(C_{max}) \tag{1}$$

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

$$\sum_{i\in T(t)} \omega_{im} \le 1, \quad m \in M, \quad t \ge 0 \tag{3}$$

$$O_i \ge 0, \qquad i = 0, 1, 2, \ldots, n \times m + 1 \tag{4}$$

where ω_{im} is the weight of setting for Operation i initiating Machine m and it also can be set to the probabilities for diversity-enhanced solutions (Wang et al. 2013). The constraint on precedence relationships is defined by Eq. (2). The constraint on one machine can process at most one operation at a time that is indicated in Eq. (3). The completion time of operations must be positive about the constraint as indicated in Eq. (4).

Parallelized bat algorithm with a communication strategy

In the parallel structure, several groups are created by dividing the population into subpopulations to construct the parallel processing. Each of the subpopulations evolves independently in regular iterations. A *good* solution based on the best fitness evaluation is selected to continue next periods. After a communication scheme is triggered, bad areas of the solution space are eliminated and exploration of promising regions is carried out. The bat-inspired algorithm should be reviewed briefly as follows the subsection, before analyzing and designing the communication strategy.

Bat-inspired algorithm

Original BA (Yang 2010) simulates parts of the echolocation characteristics of the micro-bat in a simple way. Three major characteristics of the micro-bat are employed to construct the basic structure of BA. The micro-bat, one of species of the bat is a famous example of extensively using the echolocation. Hence, the first characteristic is the echolocation behavior. The second characteristic is the frequency that the micro-bat sends out. The frequency f with a variable wavelength λ and the third characteristic is the loudness that the micro-bat uses to search for prey. The basic concepts of BAs can be described as follows:

1. Bats fly randomly with velocity v_i at position x_i. They can adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r ∈ [0, 1], depending on the proximity of their target.
2. There are many ways to adjust the loudness. For simplicity, the loudness is assumed to be varied from a positive large A_0 to a minimum constant value, which is denoted by A_{min}.

The movement of the virtual bat is simulated as follows:

$$f_i = f_{min} + (f_{max} - f_{min}) * \beta \tag{5}$$

$$v_i^t = v_i^{t-1} + \left(x_i^{t-1} - x_{best}\right) * f_i \tag{6}$$

$$x_i^t = x_i^{t-1} + \mathrm{v}_\mathrm{i}^\mathrm{t} \tag{7}$$

where, f is the frequency used by the bat seeking for its prey; f_{min} and f_{max}, represent the minimum and maximum value, respectively. x_i denotes the location of the *i-th* bat in the solution space; v_i represents the velocity of the bat, t indicates the current iteration, β is a random vector, which is drawn from a uniform distribution, $\beta \in [0, 1]$; and x_{best} indicates the global near best solution found so far over the entire population. In addition, the rate of the pulse emission from the bat is considered in the process. The micro-bat emits the echo and adjusts the wavelength depending on the proximity of their target. The pulse emission rate is denoted by the symbol r_i, and ∈ [0, 1], where the suffix i indicates the *i-th* bat. In every iteration, a random number is generated and compared with r_i. If the random number is greater than r_i, a local search strategy, namely, random walk is detonated. A new solution for the bat is generated by Eq. (8):

$$x_{new} = x_{old} + \varepsilon A^t \tag{8}$$

where ε is a random number and $\varepsilon \in [-1, 1]$; it represents the average loudness of all bats at the current time step. After updating the positions of the bats, the loudness A_i and the pulse emission rate r_i are updated only when the global near best solution is updated and the randomly generated number is smaller than A_i. The updates of A_i and r_i are operated by Eqs. (9) and (10):

$$A_i^{t+1} = \alpha A_i^t \tag{9}$$

$$r_i^{t+1} = r_i^0 \left[1 - e^{-\gamma t}\right] \tag{10}$$

where, α and γ are constants.

Communication strategy

In our previous work (Tsai et al. 2014), has proven that the parallel approach to BA is the faster and more accuracy than original one by providing the communication strategies for processing parallel of the swarm of bats in different subgroups. The swarm of bats in BA is divided into G subgroups. Each subgroup evolves from the BA optimization independently, i.e., each subgroup has its own bats and the finest solution. These finest bats among all the bats in a group will be migrated to other groups to replace the poorer bats of that group and update for each group after running every fixed number of iterations. Several partitions could be run in parallel and then merged. The child processes are halted and their results are compared. After communicating, bad areas of the solution space are eliminated and the promising regions are explored. Let G_j be the number of agents of the subgroup, where j is the index of the subgroup. While $t \cap R \neq \phi$, where t is current iteration, and R is the fixed iterations, k agents (where the top k fitness in G_j) will be copied to $G(j+1)$ to replace the same number of agents with the worst fitness, where $j = 0, 1, 2, \ldots, G$. The diagram of the parallelized BA with a communication strategy is shown in Fig. 1.

The steps can be described as follows:

1. **Initialization:** Generate bat population and divide it into G subgroups. Each subgroup is initialized by BA independently. Assign R-*th* number of iterations for executing X_{ij}^T solutions for N_j bats in the *j-th* group, $i = 0, 1, \ldots, N_{j-1}$; $j = 0, 1, \ldots, G-1$, where G is the

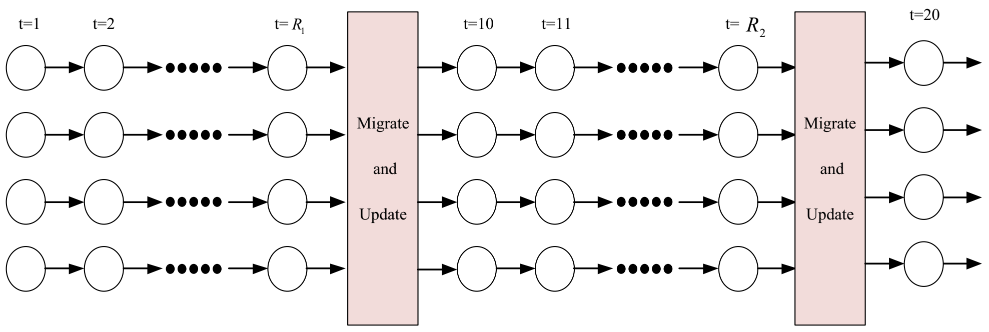

Fig. 1 The diagram of PBA with a communication strategy

number of groups, Nj is the j-th subpopulation size and t is the current iteration and set to 1.

2. **Evaluation:** Evaluate the value of $f(X_{ij}^t)$ for bats in j-th group.
3. **Update:** Update the velocity and bat positions using Eqs. (5), (6) and (7).
4. **Communication strategy:** Migrate k best bats among G_j^t to the $(j+1)$-th group G_{j+1}^t, mutate G_{j+1}^t by replacing k poorer bats in that group and update all of the group in each R iterations.
5. **Termination:** Repeat Step 2 to Step 5 until the predefined value of the function is achieved or the maximum number of iterations has been reached. Record the best value of the function $f(X_{ij}^t)$ and the best bat position among all the bats X_{ij}^t.

Parallel bat algorithm for JSSP

JSSP is a combinatorial problem, and its solution space is discrete. However, the encoding scheme for PBA is continuous, so it cannot be applied to solve JSSP directly. In order to apply PBA to JSSP, a direct mapping relationship between the job sequence and the vector of individuals in PBA must be constructed as a discrete one. To do so, a random-key encoding scheme (RES; Bean 1994) is used to combine with PBA to solve this issue. The parallel processing with communication strategy and the embedding of the random walk of a local search strategy in proposed algorithm PBA is an effective way to obtain a more effective solution. The following example illustrates the JSSP problem with three jobs ($n = 3$) and three machines ($m = 3$). The operations must be processed for jobs in Table 1. The initiated machine order for each operation and the processing times are provided in Tables 2 and 3.

The satisfied operation ordering with the stated constraint Eqs. (1–4) is a feasible schedule. The feasible result from this JSSP is 1,4,7,5,2,8,3,6,9 with the obtained makespan of 17. Figure 2 shows the Gantt chart of the schedule of this 3 × 3JSSP.

Table 1 Jobs and operation index in a 3 × 3 JSSP

Jobs	Operations		
J1	O1	O2	O3
J2	O4	O5	O6
J3	O7	O8	O9

Table 2 Machine allocation in a 3 × 3 JSSP

Machine allocations		
M1	M2	M3
O1	O3	O2
O5	O4	O6
O8	O9	O7

Table 3 Operation times in 3 × 3 JSSP

Jobs operation times			
J1 (O1,O2,O3)	2	3	5
J2 (O4,O5,O6)	1	5	2
J3 (O7,O8,O9)	4	6	4

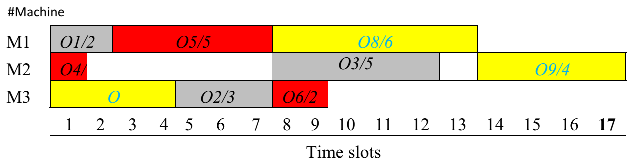

Fig. 2 A sample Gantt chart for feasible schedule for example of a 3 × 3 JSSP

Representation of individuals in PBA for JSSP

Random key encoding scheme can be used to transform a position in a continuous space to a discrete space. The searching space is created in an $n \times m$ dimensions space for n jobs on m machines JSSP. The location of a bat consists of $n \times m$ dimensions and is represented with $n \times m$ real numbers. In order to simulate an operation permutation sequence of JSSP, the $n \times m$ real numbers are transformed into an integer series from 1 to $n \times m$ by the RES. Each integer

An artificial bat	**10.2**	**2.5**	**7.5**	**6.7**	**1.3**	**5.3**	**4.2**	**8.1**	**9.5**
Integer series τ	9	2	6	5	1	4	3	7	8
$1 + \tau \bmod 3$	1	3	1	3	2	2	1	2	3
Operation sequence	o_{11}	o_{31}	o_{12}	o_{32}	o_{21}	o_{22}	o_{13}	o_{23}	o_{33}

Fig. 3 The RES processing for an individual representation in a 3 × 3 JSSP

number represents an operation index according to its ordering in all $n \times m$ real numbers. A real vector in the RES is sorted through an ascending order for an integer series $(\tau_1, \tau_2, \ldots, \tau_k)$ in which each integer τ_k indirectly represents an operation order of a job, where $1 \le k \le n \times m$. Because each job must go through m machines to complete its work, a job must have m operations that are scheduled for a preceding constraint. For this constraint, the job index can be calculated by $(1 + \tau_k \, mod \, n)$, where n is the number of jobs. An operation sequence is figured out by the integer series transformation. Scanning from the left to the right, each job index has m occurrences, which correspond to the number of operations for a job. The operation order can be satisfied with the preceding constraint because the operation permutation is feasible.

Figure 3 illustrates an example of processing from an RES virtual space to a feasible operation permutation in the 3 × 3 JSSP solution space. It is supposed that the location of a bat in an RES virtual space is (10.2, 2.5, 7.5, 6.7, 1.3, 5.3, 4.2, 8.1, 9.5). It can be encoded to an integer series (9, 2, 6, 5, 1, 4, 3, 7, 8) by sorting the 3 × 3 real numbers in an ascending order, i.e., 1.3 is the smallest number, it is then ranked to 1 and so on. In this integer series, the integers 9, 6 and 3 indicate that the operations belong to Job 1 because (9 mod 3)+1 = 1, (6 mod 3)+1 = 1 and (3 mod 3)+1 = 1. The integers 1, 4 and 7 indicate that the operations belong to Job 2, because (1 mod 3) + 1 = 2, (4 mod 3) + 1 = 2 and (7 mod 3) + 1 = 2. The integers 2, 5 and 8 indicate that the operations belong to Job 3, because (2 mod 3) + 1 = 3, (5 mod 3) + 1 = 3 and (8 mod 3) + 1 = 3, respectively. An operation permutation (1, 3, 1, 3, 2, 2, 1, 2, 3) corresponding to job indexes is obtained. Scanning from the left to the right for these operations, the first 1 means the first operation of job 1, corresponding to o_{11}, the second 1 means the second operation of job 1, corresponding to o_{12} and the third 1 means the third operation of job 1, corresponding to o_{13}. According to this scanning process, the partial series (2, 2, 2) corresponds to (o_{21}, o_{22}, o_{23}) and (3, 3, 3) corresponds to $(o_{31}, o_{32,} o_{33})$. After scanning the job index series from the left to the right, the permutation (1, 3, 1, 3, 2, 2, 1, 2, 3) corresponds to an operation sequence $(o_{11}, o_{31}, o_{12}, o_{32}, o_{21}, o_{22}, o_{31}, o_{23}, o_{33})$ as shown in Fig. 3. The operation sequence represented by this encoding scheme is always a feasible solution of JSSP.

Neighborhood operators

Because the solutions space of sequential operations in JSSP are discrete, PBA must be constructed in the discrete solutions based on the mapping relationship between the job sequence and the vector of individuals. For improving the diversity of population, enhance the quality of the solution and process parallel strategies, several operations have to be used the application methods of the diversity-enhanced solutions (Wang et al. 2013) and a wrapper feature selection (Rodrigues et al. 2014). These operations include swapping, insertion, inversion and long distance communication schemes based on Hamming distance methods. They could be defined as follows.

Swap is to choose two different positions of an operation permutation sequence randomly and swap them.

Insert is to choose two different positions of an operation permutation sequence randomly and insert the back one before the front.

Inverse is to inverse the subsequences between two different random positions of an operation permutation sequence.

Long distance communicate is to choose a subsequence in a random interval of another random the operation permutation sequence and replace the corresponding part of the subsequence.

For example, when two positions of an operation sequence in an individual need to exchange the order, the "swap" operator has to be used. Compare the makespan before being exchanged and after being exchanged, if the makespan after being exchanged is better, then the new permutation operation of this individual of the current solution will be updated.

Communication strategies

As mentioned above, the parallelization of algorithms is based on dividing (partitioning) the problem so that several partitions could be run in parallel and then merged. This idea is to run several processes for the entire problem in parallel and periodically compare the results. The search is then continued by all the processes from a common good solution. A main process reads the problem definition. A set of child processes on separate processors is then created for running an operation sequence of each machine by the communication schemes. After running specified time interval, the child processes are halted and the main process compares their results. A good solution for the child process is selected to continue next specified time interval. The promising area of the solution space will be explored. To be satisfied the constraint in Eq. (3), a machine can process only one operation at a time, total of the distributed weights for the operation diversity of individual is equal to or less than one. To do so, a solution enhancement algorithm based on the neigh-

Fig. 4 The pseudo-code of communication strategy (COM scheme)

```
Input: $sol$, the individual to communicate
Output: $sol'$, one individual after executing a communication
1: $b \leftarrow rand()$
2: if $(0 \le b \le w_s)$ then execute swapping scheme for individual sol
3: else if $(w_s \le b \le w_s + w_i)$ then execute inserting scheme for individual sol
4: else if $(w_s + w_i \le b \le w_s + w_i + w_{inv})$ then execute inversion scheme for individual sol
   //Finally, (b will match with $w_{long}$)
5:   else execute long distance communication movement scheme for individual sol
6: end if
```

```
Input: $sol$, the individual to communicate; a starting temperature $T$; a final temperature $T_f$; a cooling rate $\beta$
Output: an improvement makespan
1: Makespan $(sol) \leftarrow$ makespan of an operation permutation represented by sol
2: while $(T > T_f)$ do
3: Randomly select an operation from Communication strategy scheme (Figure 4), and generate a new individual $sol_0$ by the selected operation.
4: Makespan$(sol_0) \leftarrow$ makespan of an operation permutation represented by $sol_0$
5: $\Delta \leftarrow$ Makespan$(sol_0)$ – Makespan$(sol)$
6: if $\Delta > 0$ then   //$sol_0$ is worse than $(sol)$
// randomly generate a probability rand() to accept the worse $(sol_0)$ with a probability, $exp^{\Delta/T}$
7: if ( $R$ = rand() < min {1; , $exp^{\Delta/T}$} ) then
8: $sol \leftarrow sol_0$          //update sol to be an improved $sol_0$
9: Makespan$(sol) \leftarrow$ Makespan $(sol_0)$
10: end if
11: else
12: $sol \leftarrow sol_0$   // to accept a better $sol_0$
13: Makespan$(sol)$ $\leftarrow$ Makespan$(sol_0)$
14: $T \leftarrow \beta \times T$
15: end if
16: end while
```

Fig. 5 The pseudo-code of the objective function (CMX method)

```
Input: The probability $W_{im}$ to execute COM scheme; Objective function $f(sol)$, based on CMX method
Define pulse frequency $f_i$ at solution $_i$; initialize pulse rates $r_i$ and the loudness $A_i$
Output: one best operation permutation schedule represented by the global best
1: Initialize the bat population $sol_i$ (i = 1, 2, ..., n) and $v_i$ based on RES method
2: while the stop condition (the optimal solution is found or
          the maximal number of iteration is reached) is not met do
3: Generate new solution by RES method.
4: for each bat $_{id}$ do
5:      if $(rand > r_i)$
6:      Select a solution among the best solutions and update as Eqs. (6) and (7)
7:      Generate a local solution around the selected best solution by COM method
8:      if (rand < $W_{im}$ & rand <$A_i$)
9:      Execute the CMX method for $bat_{id}$
10:     Increase $r_i$ and reduce $A_i$, according to Eqs. (9) and (10)
11:     end if; end if
12: end for
13      Find the current best sol
14: end while
```

Fig. 6 The pseudo-code of PBA for JSSP

borhood operations, (named as COM) is employed as shown in Fig. 4. w_s, w_i, w_{inv} and w_{long} means the probability of executing the swapping scheme, the insertion scheme, the inversion scheme and the long distance movement scheme, respectively.

Objective function

The objective function is to minimize makespan with satisfied constrains in Eqs. (1–4). Evaluation for this objective function is employed in the transformed solution space of

Table 4 Computational result comparison of PBA, PSO and BA methods for instances FT and LA of the test benchmark

Instances	Size $(n \times m)$	BKS (Best known solution)	PBA (Proposed in this paper)		PSO (Ge et al. 2008)		BA (Encoded RES)	
			Best	RD	Best	RD	Best	RD
FT06	6 × 6	55	55	0.00	55	0.00	55	0.00
FT10	10 × 10	930	930	0.00	930	0.00	951	2.26
FT20	20 × 5	1165	1165	0.00	1165	0.00	1177	1.03
LA01	10 × 5	666	666	0.00	666	0.00	666	0.00
LA02	10 × 5	655	655	0.00	655	0.00	657	0.31
LA03	10 × 5	597	597	0.00	597	0.00	599	0.34
LA04	10 × 5	590	590	0.00	590	0.00	590	0.00
LA05	10 × 5	593	593	0.00	593	0.00	597	0.67
LA06	15 × 5	926	926	0.00	926	0.00	926	0.00
LA07	15 × 5	890	890	0.00	890	0.00	899	1.01
LA08	15 × 5	863	863	0.00	863	0.00	863	0.00
LA09	15 × 5	951	951	0.00	951	0.00	951	0.00
LA10	15 × 5	958	958	0.00	958	0.00	958	0.00
LA11	20 × 5	1222	1222	0.00	1222	0.00	1232	0.82
LA12	20 × 5	1039	1039	0.00	1039	0.00	1049	0.96
LA13	20 × 5	1150	1150	0.00	1150	0.00	1160	0.87
LA14	20 × 5	1292	1292	0.00	1292	0.00	1299	0.54
LA15	20 × 5	1207	1207	0.00	1207	0.00	1217	0.83
LA16	10 × 10	945	945	0.00	945	0.00	965	2.12
LA17	10 × 10	784	784	0.00	784	0.00	794	1.28
LA18	10 × 10	848	848	0.00	848	0.00	858	1.18
LA19	10 × 10	842	842	0.00	842	0.00	852	1.19
LA20	10 × 10	902	902	0.00	902	0.00	912	1.11
LA21	15 × 10	1046	1046	0.00	1046	0.00	1066	1.91
LA22	15 × 10	927	933	0.65	**932**	0.54	944	1.83
LA23	15 × 10	1032	1032	0.00	1032	0.00	1042	0.97
LA24	15 × 10	935	**941**	0.64	950	1.60	970	3.74
LA25	15 × 10	977	**977**	0.00	979	0.20	989	1.23
LA26	20 × 10	1218	1218	0.00	1218	0.00	1228	0.82
LA27	20 × 10	1235	**1247**	0.97	1256	1.70	1256	1.70
LA28	20 × 10	1216	**1216**	0.00	1227	0.90	1227	0.90
LA29	20 × 10	1152	**1179**	2.34	1184	2.78	1184	2.78
LA30	20 × 10	1355	1355	0.00	1355	0.00	1365	0.74
LA31	30 × 10	1784	1784	0.00	1784	0.00	1794	0.56
LA32	30 × 10	1850	1850	0.00	1850	0.00	1871	1.14
LA33	30 × 10	1719	1719	0.00	1719	0.00	1739	1.16
LA34	30 × 10	1721	1724	0.17	**1723**	0.12	1731	0.58
LA35	30 × 10	1888	1889	0.05	**1888**	0.00	1919	1.64
LA36	15 × 15	1268	**1279**	0.87	1281	1.03	1291	1.81
LA37	15 × 15	1397	**1411**	1.00	1415	1.72	1425	2.00
LA38	15 × 15	1196	**1208**	1.00	1213	1.42	1223	2.26
LA39	15 × 15	1233	**1236**	0.24	1246	1.05	1256	1.87
LA40	15 × 15	1222	**1225**	0.25	1240	1.47	1252	2.45

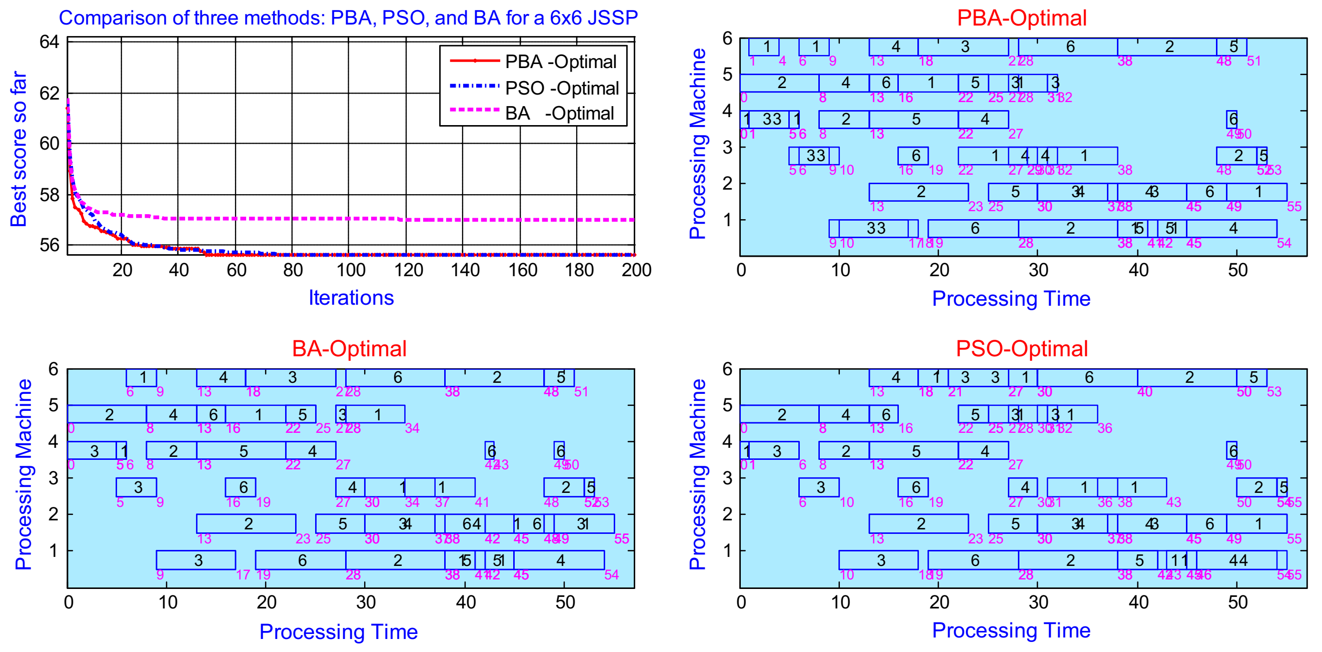

Fig. 7 Convergence comparison of BA, PSO and PBA methods for a 6 × 6 JSSP and their Gantt

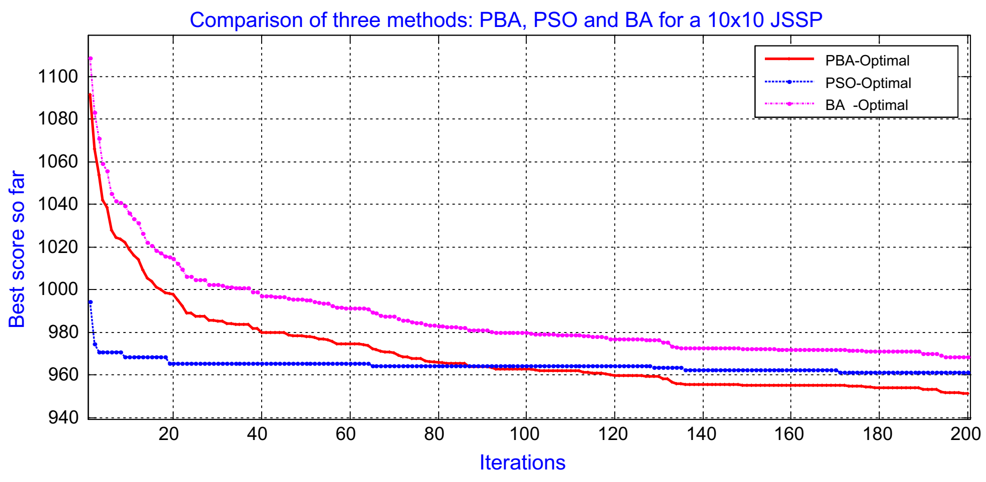

Fig. 8 Convergence comparison of BA, PSO and PBA methods for a 10 × 10 JSSP

JSSP by applying the SA algorithm (Kirkpatrick 1983). The SA algorithm has been successfully applied to many combinatorial optimization problems (Koulamas et al. 1994). The key function of SA is to allow occasional alternations to accept worsened solutions in order to increase the probability of jumping away from a local optimum and obtaining a better solution. In an SA algorithm, a new state s' is accepted with a given probability min $\{1, exp^{\Delta/T}\}$ if Δ is greater than or equal to zero, where T is a control parameter referred as temperature, $\Delta = f(s') - f(s)$, and $f()$ is objective function. The temperature T is defined by the user, and T is decreased iteration by iteration according to a referred cooling schedule from high to low. The SA algorithm is executed from high temperature until T is lower than a user-defined final temperature T_f which is a value near to zero. The individual's position can be accepted as a new position of the individual if one random probability is greater than min $\{1, exp^{\Delta/T}\}$. It means the makespan can be made improvement; otherwise, the previous position for the individual could be kept. Figure 5 shows the pseudo code for evaluating the object function namely the CMX method.

Discrete PBA for JSSP

Discrete PBA for JSSP is based on the integration of the RES, communication strategy scheme (COM) and makespan

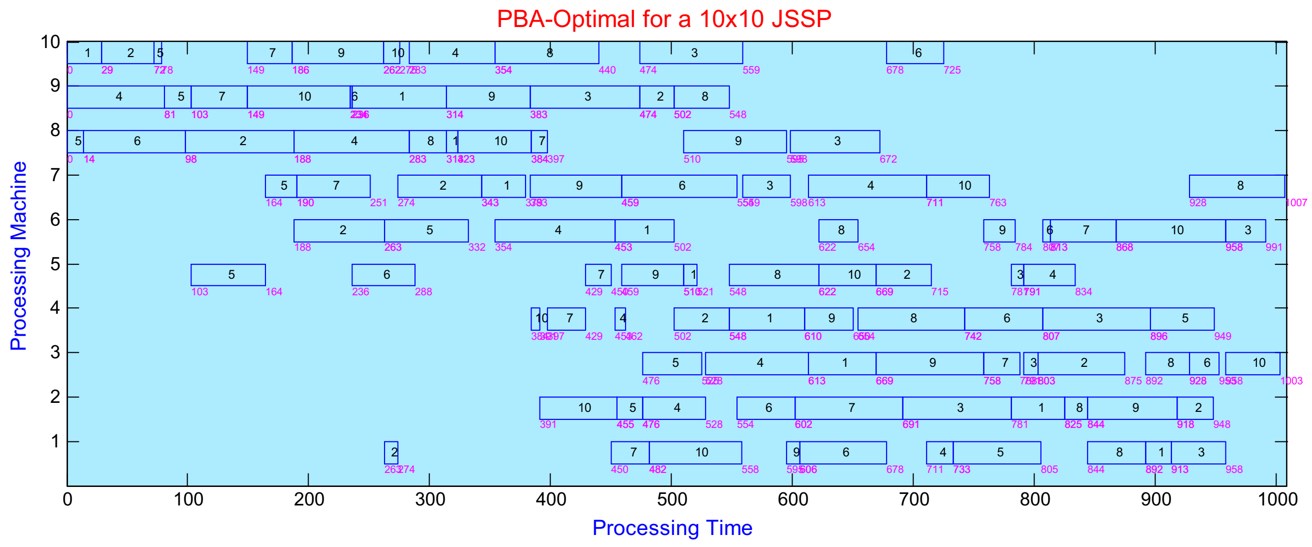

Fig. 9 The Gantt of PBA method for a 10 × 10 JSSP

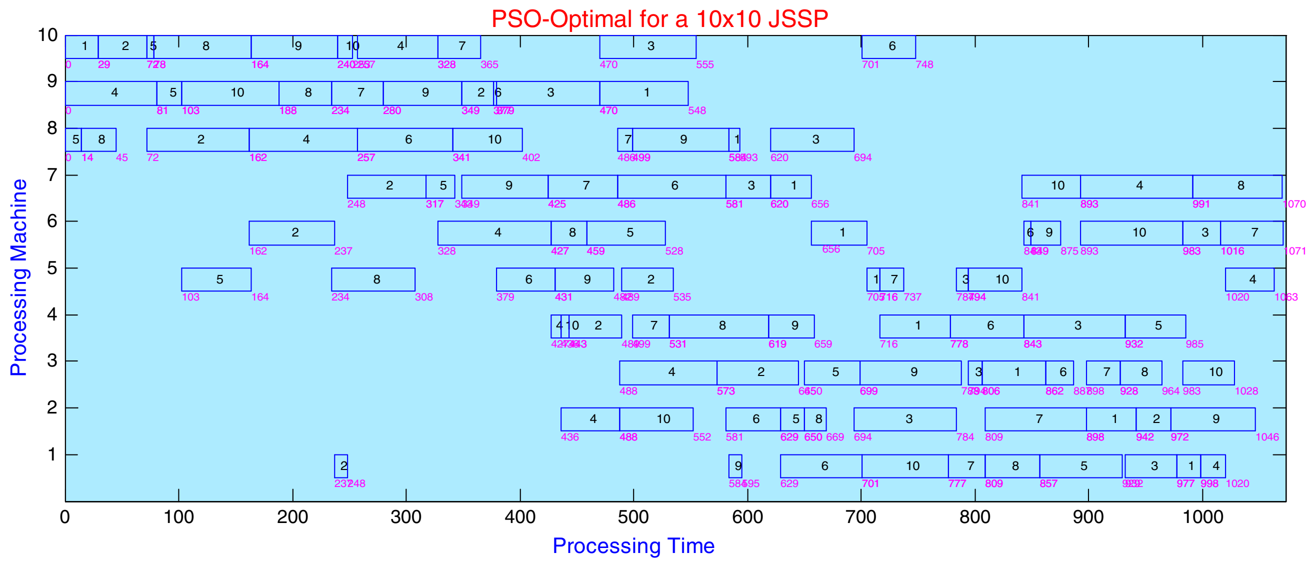

Fig. 10 The Gantt of PSO method for a 10 × 10 JSSP

(CMX) into the BA. In this method, an artificial bat is represented by a real vector as shown in row 1 of Fig. 3. Every bat flies its location in the RES space by Eqs. (6) and (7), and the objective function of one bat corresponding to the solution space of JSSP can be evaluated by the transformation from RES space to a solution space of JSSP by Eqs. (1–4). For the communication strategy of parallel processing, bad areas of the solution space are eliminated and exploration of promising regions is carried out. The RES encoding scheme provides a search space for the continuous PBA and an easy way to encode the representation of artificial bat in PBA. The weight distribution in COM communication strategy scheme provides a selected bat, which can be in a better location than the previous one. Then, each bat can fly to a new location according to Eqs. (6), (7) and (8). The process of objective function CMX scheme and PBA are executed until it obtains the optimal solution or the maximum iteration number is reached (Fig. 6).

Experimental results

The performance of the proposed method of the PBA for the JSSP is examined by using some test problems taken from the OR- Library (Beasley 1990) as the test benchmarks. Four three instances from two classes of standard JSSP test prob-

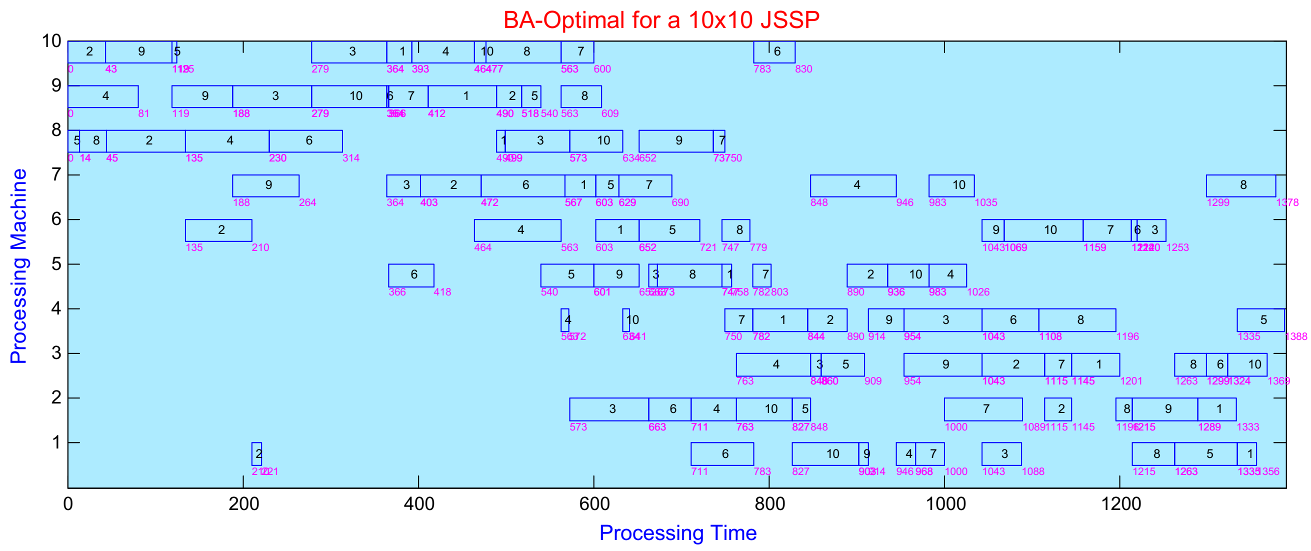

Fig. 11 The Gantt of BA method for a 10 × 10 JSSP

Table 5 Average experimental result comparison of PBA, PSO and BA methods

Instances	BKS	PBA			PSO			BA		
		Best	Avg	ARD	Best	Avg	ARD	Best	Avg	ARD
FT06.06 × 06	**55**	**55**	55.0	0.00	**55**	55.0	0.00	**55**	55.0	0.00
FT10.10 × 10	**930**	**930**	930.0	0.00	**930**	937.0	0.75	951	951.0	2.26
FT20.20 × 05	**1165**	**1165**	1165.0	0.00	**1165**	1173.0	0.69	1177	1177.0	1.03
LA01.10 × 05	**666**	**666**	666.2	0.03	**666**	666.0	0.00	**666**	666.8	0.12
LA06.15 × 05	**926**	**926**	926.6	0.06	**926**	926.6	0.06	**926**	926.0	0.00
LA11.20 × 05	**1222**	**1222**	1222.1	0.01	**1222**	1222.3	0.02	1232	1234.0	0.98
LA16.10 × 10	**945**	**945**	945.2	0.02	**945**	945.2	0.02	965	966.0	2.22
LA21.15 × 10	**1046**	**1046**	1046.8	0.08	**1046**	1053.8	0.75	1066	1069.0	2.20
LA26.20 × 10	**1218**	**1218**	1223.0	0.41	**1218**	1222.0	0.33	1228	1228.0	0.82
LA31.30 × 10	**1784**	**1784**	1790.2	0.35	**1784**	1786.0	0.11	1794	1810.8	1.50
LA36.15 × 15	**1268**	1279	1280.0	0.95	1281	1288.0	1.58	1291	1289.4	1.69

lems included Fisher and Thompson (Fisher and Thompson 1963) with instances FT06, FT10, and FT20, and Lawerence (Lawler et al. 1993) with instances LA01–LA40.

The parameters setting for both PBA and BA as referring to (Tsai et al. 2014); the maximum of location is limited to $n \times m$ and the population size of the swarm is set to 30. The initial weight for COM scheme for the entire procedure of the PBA for the JSSP algorithm is set to be 0.01. The initial temperature T is set to be the difference between the makespan of the selected bat and the best known solution, T_f is set to be 0.1 and β is set to be 0.97. Each instance contains the full 200 iterations. The evaluated experimental results compared with the obtained using the BA and PSO (Ge et al. 2008) methods are listed in Table 4. The Gantts chart and convergences curves of the first two instances are illustrated in Figs. 7, 8, 9, 10 and 11.

Figure 7 shows the convergence rate of the first instances FT06 with size 6 × 6 and their plot Gantts of the three methods, namely PBA, PSO and BA for JSSP. Figures 8, 9, 10 and 11 show the convergence rate of the second instances FT10 with size 10 × 10 JSSP and their plot Gantts of the PBA, PSO and BA methods. It is clear that the proposed PBA method converges faster than the of BA and PSO methods.

In Table 4, *instance* means the problem name, *size* means the problem size n jobs on m machines, *BKS* means the best known solution for the instance (Muth and Thompson 1963), *Best* means the best solution found by each algorithm, and *RD* means the percentage of the deviation with respect to the best known solution for the proposed method and PSO method, namely HIA (Ge et al. 2008). The boldface in Table 4 represents the better solution for one instance between PBA and PSO methods. According to the best known solutions presented in Table 4, the PBA method outperforms the BA method in 36 instances and outperforms PSO method in 10 instances. However, the PBA method performs slightly an inferior the PSO method in three other instances. In general, the PBA method can obtain the optimal area in the search space with diversity, and can get better solution than the PSO and BA methods for the JSSP scheduling problems.

The comparisons of the PBA, PSO and BA methods for testing the stable and diversity by averaging experimental results of five runs was shown in Table 5. Each run is executed 200 iterations. Instances FT06, FT10, FT20 and the first instance of other type instance set are selected as test benchmark. BKS and Best are the same meaning as those in Table 4. ARD means the relative deviation of the average solution and Avg the average of results for 5 runs, respectively. The boldface in Table 5 represents the obtained solutions for instances of the used methods PBA, PSO and BA could have been reached the BKS. Observing Table 5, the difference between the Best and the BKS, and the difference between the Avg and the BKS are within 1% and 2% for PBA and PSO methods respectively. However, this figure is higher than 2% for BA method. Because of the BA method is easy to be trapped in a local optimal and cannot find a better solution. Above experimental results show that the proposed method can obtain the better solutions than those obtained of the PSO and BA methods by providing the diversity-enhanced bats to speed up solutions.

Conclusion

In this paper, a solution to the NP-hard job shop scheduling problems (JSSP) based on parallel versions of bat algorithm (PBA), RES, communicating strategy schemes (COM) and makespan scheme (CMX) was presented. The aim at the parallel BA with communication strategies is to correlate individuals in swarms and to share the computation load among numerous processors. In this situation, the entire problem has several partitions that could be run in parallel and then merged. After running specified time intervals, the child processes are halted and their results compared in the main process. A *good* solution for the child process is selected to continue with. A good solution might be the one with the best fitness. After triggering communication schemes, bad areas within the solution space are eliminated and the exploration of promising region is carried out. In the proposed method, a location of a bat composed of $n \times m$ real numbers can be represented in the discrete solution space of JSSP by the encoding RES scheme. A speed up of selection solutions under the constraint is figured out by the COM schemes. The objective function of the bat corresponding to the solution space of JSSP could be evaluated by the application the SA algorithm to minimize the makespan. In the experiments, variety of sizes in instances of job shop scheduling benchmarks in the literature were used to test the behavior of convergence, the accuracy, and the speed of the proposed method. The results were compared with those obtained using the BA and PSO methods. The comparison indicates that the proposed method provides competitive results.

References

Abramson, D., & Abela, J. (1991). A parallel genetic algorithm for solving the school timetabling problem. In *Proceedings of the appeared in 15 Australian computer ccience conference*, (p. 10). Hobart, Australia.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, *6*(2), 154–160.

Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 1069–1072.

Behnamian, J., & Fatemi Ghomi, S. M. T. (2014). A survey of multi-factory scheduling. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-014-0890-y.

Błażewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, *93*(1), 1–33.

Çakar, T. (2011). Single machine scheduling with unequal release date using neuro-dominance rule. *Journal of Intelligent Manufacturing*, *22*(4), 481–490. doi:10.1007/s10845-009-0309-3.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. *Computers and Industrial Engineering*, *30*(4), 983–997. doi:10.1016/0360-8352(96)00047-2.

Chu, S. C., Roddick, J. F., & Pan, J.-S. (2004). Ant colony system with communication strategies. *Information Sciences*, *167*(1–4), 63–76. doi:10.1016/j.ins.2003.10.013.

Chu, S. C., Roddick, J. F., & Pan, J.-S. (2005). A parallel particle swarm optimization algorithm with communication strategies. *Journal of Information Science and Engineering*, *21*(4), 9.

Coello, C. A. C., Rivera, D. C., & Cortes, N. C. (2003). Use of an artificial immune system for job shop scheduling. In *Artificial immune systems* (pp. 1–10). Springer.

Davis, L. (1985). Job shop scheduling with genetic algorithms. In *Proceedings of an international conference on genetic algorithms and their applications*, (Vol. 140). Pittsburgh, PA: Carnegie-Mellon University.

Dell'Amico, M., & Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, *41*(3), 231–252.

Fisher, H., & Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In *Industrial Scheduling*, (Vol. 3). New Jersey: Prentice-Hall.

Garey, M. R., & Johnson, D. S. (1990). *Computers and intractability; A guide to the theory of NP-completeness*. New York, NY: W. H. Freeman & Co.

Ge, H.-W., Sun, L., Liang, Y.-C., & Qian, F. (2008). An effective PSO and AIS-based hybrid intelligent algorithm for Job-shop scheduling. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *38*(2), 358–368. doi:10.1109/TSMCA.2007.914753.

Gen, M., & Lin, L. (2014). Multiobjective evolutionary algorithm for manufacturing scheduling problems: State-of-the-art survey. *Journal of Intelligent Manufacturing*, *25*(5), 849–866. doi:10.1007/s10845-013-0804-4.

Geyik, F., & Cedimoglu, I. (2004). The strategies and parameters of tabu search for job-shop scheduling. *Journal of Intelligent Manufacturing*, *15*(4), 439–448. doi:10.1023/B:JIMS.0000034106.86434.46.

Gonçalves, J. F., Mendes, J. J. d. M., & Resende, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, *167*(1), 77–95.

Hosseini, S., & Al Khaled, A. (2014). A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. *Applied Soft Computing*, *24*, 1078–1094.

Hosseini, S., Khaled, A., & Vadlamani, S. (2014). Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. *Neural Computing and Applications*, *25*(7–8), 1871–1885. doi:10.1007/s00521-014-1678-x.

Kirkpatrick, S. (1983). Optimization by simmulated annealing. *Science*, *220*(4598), 671–680.

Koulamas, C., Antony, S., & Jaen, R. (1994). A survey of simulated annealing applications to operations research problems. *Omega*, *22*(1), 41–56.

Kuck, D. J. (1977). A survey of parallel machine organization and programming. *ACM Computing Surveys (CSUR)*, *9*(1), 29–59.

Lawler, E. L., Lenstra, J. K., Kan, A. H. R., & Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations Research and Management Science*, *4*, 445–522.

Lian, Z., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied Mathematics and Computation*, *183*(2), 1008–1017. doi:10.1016/j.amc.2006.05.168.

Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., et al. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, *37*(3), 2629–2636. doi:10.1016/j.eswa.2009.08.015.

Luh, G.-C., & Chueh, C.-H. (2009). A multi-modal immune algorithm for the job-shop scheduling problem. *Information Sciences*, *179*(10), 1516–1532. doi:10.1016/j.ins.2008.11.029.

Meeran, S., & Morshed, M. S. (2012). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: A case study. *Journal of Intelligent Manufacturing*, *23*(4), 1063–1078. doi:10.1007/s10845-011-0520-x.

Mirabi, M., Ghomi, S. M. T. F., & Jolai, F. (2013). A two-stage hybrid flowshop scheduling problem in machine breakdown condition. *Journal of Intelligent Manufacturing*, *24*(1), 193–199. doi:10.1007/s10845-011-0553-1.

Moin, N. H., Chung Sin, O., & Omar, M. (2015). Hybrid genetic algorithm with multiparents crossover for job shop scheduling problems. *Mathematical Problems in Engineering*, *2015*, 12. doi:10.1155/2015/210680.

Muth, J. F., & Thompson, G. L. (1963). *Industrial scheduling*. New Jersey: Prentice-Hall.

Qiu, X., & Lau, H. K. (2014). An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing*, *25*(3), 489–503. doi:10.1007/s10845-012-0701-2.

Rodrigues, D., Pereira, L. A. M., Nakamura, R. Y. M., Costa, K. A. P., Yang, X.-S., Souza, A. N., et al. (2014). A wrapper approach for feature selection based on bat algorithm and optimum-path forest. *Expert Systems with Applications*, *41*(5), 2250–2258. doi:10.1016/j.eswa.2013.09.023.

Song, S. Z., Ren, J. J., & Fan, J. X. (2012). Improved simulated annealing algorithm used for job shop scheduling problems. In *Advances in electrical engineering and automation* (pp. 17–25). Springer.

Tsai, C.-F., Dao, T.-K., Yang, W.-J., Nguyen, T.-T., & Pan, T.-S. (2014). Parallelized bat algorithm with a communication strategy. In M. Ali, J.-S. Pan, S.-M. Chen, & M.-F. Horng (Eds.), *Modern advances in applied intelligence*, Lecture Notes in Computer Science. (Vol. 8481, pp. 87–95). Springer International Publishing.

Tsai, P.-W., Pan, J.-S., Chen, S.-M., Liao, B.-Y., & Hao, S.-P. (12-15 July 2008). Parallel cat swarm optimization. In *Machine learning and cybernetics, 2008 international conference on*, (Vol. 6, pp. 3328–3333). doi:10.1109/ICMLC.2008.4620980.

Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*(1), 113–125.

Wang, H., Sun, H., Li, C., Rahnamayan, S., & Pan, J.-S. (2013). Diversity enhanced particle swarm optimization with neighborhood search. *Information Sciences*, *223*, 119–135. doi:10.1016/j.ins.2012.10.012.

Whitley, D., Rana, S., & Heckendorn, R. B. (1998). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, *1305*(1997), 6.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, *1*(1), 67–82. doi:10.1109/4235.585893.

Wolpert, D. H., & Macready, W. G. (2005). Coevolutionary free lunches. *Evolutionary Computation, IEEE Transactions on*, *9*(6), 721–735. doi:10.1109/TEVC.2005.856205.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In J. González, D. Pelta, C. Cruz, G. Terrazas, & N. Krasnogor (Eds.), *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Studies in Computational Intelligence. (Vol. 284, pp. 65–74). Berlin Heidelberg: Springer.

Yang, X.-S., & He, X. (2013). Bat algorithm: Literature review and applications. *International Journal of Bio-Inspired Computation*, *5*(3), 141–149.

Yin, M., Li, X., & Zhou, J. (2011). An efficient job shop scheduling algorithm based on artificial bee colony. *Scientific Research and Essays*, *6*(12), 2578–2596.

Ying, K.-C., Lee, Z.-J., & Lin, S.-W. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, *23*(5), 1795–1803. doi:10.1007/s10845-010-0483-3.

Zhang, C., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers and Operations Research*, *34*(11), 3229–3242.

Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, *10*(1), 79–89. doi:10.1016/j.asoc.2009.06.008.