Engineering Applications of Artificial Intelligence ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Job Shop Scheduling with the Best-so-far ABC

Anan Banharnsakun, Booncharoen Sirinaovakul, Tiranee Achalakul *

Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand

ARTICLE INFO

Article history:
Received 24 January 2011
Received in revised form
31 May 2011
Accepted 7 August 2011

Keywords:
Best-so-far Artificial Bee Colony (Best-so-far ABC)
Swarm intelligence
Variable Neighboring Search (VNS)
Job Shop Scheduling Problem (JSSP)

ABSTRACT

The Job Shop Scheduling Problem (JSSP) is known as one of the most difficult scheduling problems. It is an important practical problem in the fields of production management and combinatorial optimization. Since JSSP is NP-complete, meaning that the selection of the best scheduling solution is not polynomially bounded, heuristic approaches are often considered. Inspired by the decision making capability of bee swarms in the nature, this paper proposes an effective scheduling method based on Best-so-far Artificial Bee Colony (Best-so-far ABC) for solving the JSSP. In this method, we bias the solution direction toward the Best-so-far solution rather a neighboring solution as proposed in the original ABC method. We also use the set theory to describe the mapping of our proposed method to the problem in the combinatorial optimization domain. The performance of the proposed method is then empirically assessed using 62 benchmark problems taken from the Operations Research Library (OR-Library). The solution quality is measured based on "Best", "Average", "Standard Deviation (S.D.)", and "Relative Percent Error (RPE)" of the objective value. The results demonstrate that the proposed method is able to produce higher quality solutions than the current state-of-the-art heuristic-based algorithms.

1. Introduction

The Job Shop Scheduling Problem (JSSP) is a real-world problem in a field of production management. To survive in the modern competitive marketplace, which requires lower cost and shorter product life cycles, a corporation must respond quickly and precisely to the customer's demands. Effective scheduling plays an important role in this adaptation.

JSSP is an optimization problem that can be described in terms of a set of jobs, each with one or more operations. The operations of a job have to be processed in a specified sequence on a specific set of machines. The time required for all operations to complete their processes is called the *makespan*. The objective of JSSP aims to minimize the makespan value.

Many approaches using both mathematical formulations and heuristic methods have been developed to solve this problem. For a JSSP situation of small size, mathematical formulations such as integer programming techniques (Fisher, 1981) can be used to solve this problem in a reasonable computational time. However, Garey et al. (1976) provided a proof that this problem is NP-complete, i.e. as the problem size increases, the computational time to find the best schedule using the mathematical formulation methods grows exponentially. To handle this issue, heuristic methods such as branch-and-bound (Carlier and Pinson, 1989) have been considered.

Metaheuristics (Yang, 2008) are one of many approximation methods widely used to solve practical optimization problems. In recent years, several algorithms employing a metaheuristic approach such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bee Colony Optimization (BCO), and Artificial Bee Colony (ABC) have been applied to solve JSSP.

Watanabe et al. (2005) introduced a genetic algorithm (GA) with a modified crossover operator and a search area adaptation method for controlling the tradeoff balance between global and local searches. Goncalves et al. (2005) presented a hybrid method called Hybrid Genetic Algorithm (HGA). The method combined GA and local search based on a disjunctive graph model and a neighborhood approach to improve the solution. To enhance the performance of GA, Asadzadeh and Zamanifar (2010) proposed an agent-based parallel GA approach. This parallel approach is based on a coarse-grained model. The initial population is divided into sub-populations, and each sub-population is evolved separately. Communication between sub-populations is restricted to the migration of chromosomes. Heinonen and Pettersson (2007) developed a hybrid approach based on an Ant Colony Optimization algorithm (ACO) and a post-processing algorithm to enhance the ACO performance for solving the JSSP.

To improve the solution quality in JSSP, Tasgetiren et al. (2006) presented a hybrid method (PSO-VNS) based on the Particle Swarm Optimization (PSO) and the Variable Neighboring Search

* Corresponding author.
E-mail addresses: anan_cpe@yahoo.com (A. Banharnsakun), boon@kmutt.ac.th (B. Sirinaovakul), tiranee@cpe.kmutt.ac.th (T. Achalakul).

0952-1976/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.08.003

(VNS). To further improve efficiency of PSO, a new hybrid swarm intelligence algorithm (MPSO) consisting of particle swarm optimization, Simulated Annealing (SA) and a multi-type individual enhancement scheme was developed by Lin et al. (2010). Ge et al. (2007) employed a high global search efficiency of PSO with a powerful ability to avoid being trapped in local minima of SA by introducing an algorithm called Hybrid Evolutionary Algorithm (HEA). Ge et al. (2008) exploited the capabilities of distributed and parallel computing in swarm intelligence approaches by proposing a computationally efficient algorithm for combining PSO with an Artificial Immune System (AIS) to find the minimum makespan for Job Shop Scheduling.

Most of these aforementioned approaches aim to improve their algorithm's performance by introducing a hybrid method. They combined the advantageous features of each algorithm to improve both the local search and global search capability of their algorithms.

Inspired by the decision making capability of bee swarms, Chong and Low (2006) explored an evolutionary computation based on Bee Colony Optimization (BCO) to solve JSSP. The scheduling construction in this approach was done based on a state transition rule. Yao et al. (2010) presented an Improved Artificial Bee Colony algorithm (IABC) to enhance the search performance of the original ABC for solving the JSSP. The mutation operation was also utilized in this method for exploring the search space and avoiding local optima.

In this paper, we propose a modified version of the ABC algorithm called Best-so-far ABC. The research aims to improve the solution quality, which is measured based on "Best", "Average", "Standard Deviation (S.D.)", and "Relative Percent Error (RPE)" of the objective value. The algorithm is presented and applied to solve the JSSP.

This paper is organized as follows. Section 2 introduces the Best-so-far ABC Metaheuristic algorithm. Section 3 describes the JSSP. Section 4 proposes a mapping for applying the Best-so-far ABC to the JSSP. Section 5 presents the experiments. Section 6 compares and discusses the performance results. Finally, Section 7 presents our conclusions.

2. The Best-so-far ABC Metaheuristic

The intelligent behavior of honey bees for seeking a quality food source in nature was the inspiration for the Artificial Bee Colony (ABC) algorithm presented by Karaboga (2005). The focus of this section is first on the introduction of ABC algorithm. Next, we explain the concepts of the ABC algorithm based on the Best-so-far method for solving the combinatorial optimization problems.

2.1. Bees' foraging behavior

Foraging (Biesmeijer and Seeley, 2005) is how honey bees find food sources. In this process, quality food sources are selected based on group decision making by the swarm. Independence and interdependence in collective decision making are important factors in this mechanism.

The bees independently evaluate the quality of different new candidate food sources on their own. However, the interdependence among them makes them more attentive to candidate food sources discovered and advertised by others. Waggle dances, which are done by employed bees in the food source selection process are used to exchange information on new candidate food sources and to recruit unemployed bees to follow employed bees to their sources. Through this kind of information exchange and learning, the honeybee swarm manages to discover the highest quality food sources.

2.2. ABC algorithm

ABC algorithm takes concepts from this foraging process to discover good solutions in an optimization problem. Essential components in ABC modeled after the foraging processes are defined as follows:

- ***Food Source*** represents a feasible solution in an optimization problem.
- ***Fitness Value*** represents the *profitability* of a food source. For simplicity, it is represented as a single quantity associated with an objective function of a feasible solution.
- ***Bee Agents*** is a set of computational agents. The agents in ABC are categorized into three groups: employed bees, onlooker bees, and scout bees. The colony is equally separated into employed bees and onlooker bees. Each solution in the search space consists of a set of optimization parameters, which represent a food source's "location". The number of employed bees is equal to the number of food sources, i.e. there would be one employed bee for each food source.

The employed bees will be responsible for investigating their food sources and sharing the information about these food sources to recruit the onlooker bees. The onlooker bees will make a decision to choose a food source based on this information. A food source that has a higher quality will have a larger probability of being selected by onlooker bees. An employed bee whose food source is rejected as low quality by employed and onlooker bees will change to a scout bee to search randomly for new food sources. The details of the algorithm are as follows.

First, randomly distributed initial food source positions are generated. The objective function determines how good a solution is. It can be represented by

$$F(x_i), x_i \in R^D, \quad i \in \{1,2,3\ldots,SN\} \tag{2.1}$$

x_i is the position of a food source as a D-dimensional vector, $F(x_i)$ is the objective function, and SN is the number of food sources.

After initialization, the population is subjected to repeated cycles of four major steps: updating feasible solutions by employed bees, selection of feasible solutions by onlooker bees, updating feasible solutions by onlooker bees, and avoidance of suboptimal solutions by scout bees.

2.2.1. Updating feasible solutions by employed bees

The position of the new feasible food source discovered by an employed bee is calculated from

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \tag{2.2}$$

In Eq. (2.2), v_{ij} is a new feasible solution that is modified from its previous solution value (x_{ij}) based on a comparison with the randomly selected position from its neighboring solution (x_{kj}). $\emptyset_{ij}$ is a random number between $[-1,1]$ which is used to adjust the old solution to become a new solution in the next iteration. $k \in \{1,2,3\ldots,SN\} \wedge k \neq i$ and $j \in \{1,2,3\ldots,D\}$ are randomly chosen indexes. The difference between x_{ij} and x_{kj} is a difference of position in a particular dimension.

If a new food source (v_{ij}) is better than an old food source (x_{ij}), the old food source is replaced by the new food source.

2.2.2. Selection of feasible solutions by onlooker bees

When the employed bees return to their hive, they share information with the onlooker bees about candidate solutions they found. The onlooker bees select these solutions based on

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

probability. Solutions of higher fitness have a larger chance of being selected by onlooker bees than ones of lower fitness. The probability that a food source will be selected can be obtained from

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{2.3}$$

where fit_i is the fitness value of the food source i, which is related to the objective function value ($F(x_i)$) of the food source i.

2.2.3. Updating feasible solutions by onlooker bees

Based on the information obtained from the employed bees, the onlooker bees select their feasible food sources. The selected food sources are then updated using Eq. (2.2), i.e. an old food source is replaced by a new food source if the new food source is of a better quality.

2.2.4. Avoidance of suboptimal solutions by scout bees

This step is done by reassigning employed bees whose contributions are rejected as low quality to become scout bees who will randomly search for new solutions. The new random position chosen by the scout bee will be calculated from

$$x_{ij} = x_j^{min} + rand[0,1](x_j^{max} - x_j^{min}) \tag{2.4}$$

where x_j^{min} is the lower bound of the food source position in dimension j and x_j^{max} is the upper bound of the food source position in dimension j.

These four major steps mentioned above are repeated until an optimal solution is found or the number of iteration (cycle) reaches the termination criteria, MCN (Karaboga and Basturk, 2007).

The processes of the ABC Metaheuristic can be shown in pseudo-code as Fig. 1.

In a robust search process, exploration and exploitation must be carried out together. While the exploration process is related to the independent search for an optimal solution, exploitation uses existing knowledge to bias the search space. In the mentioned ABC algorithm, the exploitation will be handled by employed and onlooker bees while the exploration will be maintained by scout bees. This mechanism enables ABC to have both the local and the global search ability. Based on this advantage, the ABC algorithm will be able to get out of a local optimum point in the search space and find the global optima better than other heuristic approaches such as Simulated Annealing and Tabu Search that only have a local search ability. As a result, many researchers have applied the ABC algorithm to solve several science and mathematical applications (Rao et al., 2008; Singh, 2009; Sabat et al., 2010). The comparison results showed that the ABC algorithm performs better than other heuristic algorithms in terms of solution quality and computation efficiency.

Although the activities of exploitation and exploration are well balanced and help to mitigate both stagnation and premature convergence in the ABC algorithm, the convergence speed is still an issue in some situations.

```
Procedure ABC_Metaheuristic
        Initial_Solutions
        While (criterion)
                Update_Feasible_Solutions (Employed bees)
                Select_Feasible_Solutions (Onlooker bees)
                Update_Feasible_Solutions (Onlooker bees)
                Avoid_Sub-Optimal_Solutions (Scout bee)
        End while
End-Procedure
```

Fig. 1. ABC Metaheuristic procedure.

2.3. Best-so-far ABC algorithm

The Best-so-far ABC is a modified version of the ABC algorithm proposed by Banharnsakun et al. (2011) to enhance the exploitation and exploration processes.

In the original algorithm, each onlooker bee selects a food source based on a probability that varies according to the fitness function. Then the new candidate food sources are generated by updating the onlooker solutions based on a single neighboring employed bee as shown in Eq. (2.2).

In the Best-so-far ABC method, all onlooker bees use the information from all employed bees to make a decision on a new candidate food source. Thus, the onlookers can compare information from all candidate sources and are able to select the best-so-far position. As a result, the Eq. (2.2) for onlooker bees is modified. The old food source will be updated to the new food source by comparing with the best-so-far food source rather than comparing with the neighboring food source. This change should make the Best-so-far ABC algorithm converge more quickly because the solution will be biased towards the best solution found so far. The new method used to calculate a new candidate food source for the onlooker bee is

$$v_{id} = x_{ij} + \Phi f_b(x_{ij} - x_{bj}) \tag{2.5}$$

where v_{id} is the new candidate food source for onlooker bee position i and dimension d, $d=1,2,3,\ldots D$; x_{ij} is the selected food source position i in a selected dimension j; Φ is the random number between -1 and 1; f_b is the fitness value of the best food source so far; x_{bj} is the position of the best-so-far source in a selected dimension j.

Although the best-so-far method can increase the local search ability compared to the original ABC algorithm, the solution is easily entrapped in a local optimum. In order to resolve this issue, an adjustable search radius for the scout bee was also introduced in the Best-so-far ABC algorithm. Based on the assumption that the food source of a scout bee will be far from the optimal food source in the first iteration and it will become closer to the optimal food source in later iterations, the scout bee in the Best-so-far ABC will randomly generate a new food source using Eq. (2.6) rather than using Eq. (2.4)

$$v_{ij} = x_{ij} + \emptyset_{ij}\left[\omega_{max} - \frac{iteration}{MCN}(\omega_{max} - \omega_{min})\right]x_{ij} \tag{2.6}$$

where v_{ij} is a new feasible solution of a scout bee that is modified from the current position of an abandoned food source (x_{ij}) and $\emptyset_{ij}$ is a random number between $[-1,1]$. The value of ω_{max} and ω_{min} represent the maximum and minimum percentage of the position adjustment for the scout bee, respectively. The value of ω_{max} and ω_{min} are fixed to 1 and 0.2, respectively. These parameters were empirically chosen by the experimenter (Banharnsakun et al., 2011). With these selected values, the adjustment of scout bee's position based on its current position will linearly decrease from 100% to 20% in each experiment round.

Similar to the original ABC, the Best-so-far ABC algorithm is designed for solving numerical optimization problems and the algorithm utilizes Eqs. (2.5) and (2.6) for finding a solution in a continuous function domain. These equations cannot be used directly to solve problems in combinatorial optimization, while its solution is in a discrete domain. However, the algorithm can be expanded for this problem type with suitable modifications (Karaboga and Akay, 2009).

There are several methods proposed to modify the ABC algorithm for solving discrete optimization problems. Singh (2009) employed a subset encoding to represent a solution in an artificial bee colony algorithm for solving the leaf-constrained minimum spanning tree problem. Pan et al. (2011) also proposed

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

a discrete artificial bee colony (DABC) algorithm to minimize total weighted earliness and tardiness penalties for the lot-streaming flow shop scheduling problems. In our work, we introduce a set theory and a discrete job permutation to represent the solution.

We consider the search process in the Best-so-far ABC Metaheuristic in terms of (S,f,Ω), where S is the set of candidate solutions, f is the fitness function, which assigns a fitness value $f(s)$ to each candidate solution $s \in S$ and Ω is a set of constraints. Solutions $\tilde{s}$ belonging to the set $\tilde{S} \subseteq S$that satisfy Ω are called feasible solutions.

In our Best-so-far ABC, the initial solutions, which are the feasible solutions $(\tilde{s})$ in the feasible search space $\tilde{S}$, are randomly constructed and assigned to employed bees. After initialization, the artificial bees are subjected to repeated cycles of three major processes: updating feasible solutions, selecting feasible solutions, and avoiding suboptimal solutions.

In the process of updating feasible solutions, employed bees update their solutions with their neighbors by using the concept of Eq. (2.2). The old solutions in an employed bee's memory are replaced with new solutions of higher fitness.

During the selection of feasible solutions $(\tilde{s})$, each onlooker bee selects one of the proposed solutions depending on the information obtained from the employed bees. The probability of a solution being selected by an onlooker bee is proportional to its fitness as calculated by Eq. (2.3). After solutions are selected, the onlooker bees also update their selected solutions using the concept of Eq. (2.5) based on the best-so-far solution $s_b \in \tilde{S}$, where $f(s_b) \geq f(\tilde{s})$ for all $\tilde{s}$ obtained from employed bees in each iteration. Next, the old solutions in an onlooker bee's memory are replaced with new solutions of higher fitness from the same step of employed bee.

In the process of avoiding suboptimal solutions, solutions that do not improve the fitness are replaced with new solutions randomly constructed by the scout bees. The concept to generate the new solution is based on the Eq. (2.6).

These three major processes are repeated until a globally optimal feasible food source $s^* \in \tilde{S}$, where $f(s^*) \geq f(\tilde{s})$ for all $\tilde{s} \in \tilde{S}$ is found or the number of iteration reaches the Maximum Cycle Number (MCN).

3. Job Shop Scheduling Problem description

French (1982) described the Job shop Scheduling Problem (JSSP) as a set of n jobs denoted by J_j where $j=1,2,\ldots, n$, which have to be processed on a set of m machines denoted by M_k, where $k=1,2,\ldots,m$. Operation of jth job on the kth machine will be denoted by O_{jk} with the processing time P_{jk}. Each job must be processed through all machines in a particular order also known as the technological constraint and processing time varies with each job. A machine can process only one job at a time. The required order of machines also varies from one job to another. Once a machine starts to process a job, no interruption is allowed. The time required for all operations to complete their processes is called makespan. The objective of JSSP aims to minimize the makespan value. A solution can be expressed as

$$MinC_{max} = \max(C_1, C_2, C_3, \ldots C_n)$$

$$C_j = \sum_{k=1}^{n} (w_{jk} + p_{jm(k)}) \tag{3.1}$$

where C_j is the completion time of job j; w_{jk} is the waiting time of job j at sequence k; and $p_{jm(k)}$ is the processing time needed by job j on machine m at sequence k

The difficulty in finding good solutions to JSSP is due to the number of feasible solutions. For large problem size, the number of possible solutions makes it nearly impossible to explore the entire solution space. The total number of all possible solutions is $(n!)^m$.

4. Mapping Best-so-far ABC algorithm to the JSSP

We applied the Best-so-far ABC method to JSSP described above. The goal is to find a global optimization of the makespan, i.e. we try to find the job operation scheduling list that minimizes the makespan value. The adopted algorithm is illustrated in Fig. 2.

The steps of operation can be described as follows:

The First Step

The initial parameters such as the number of employed bees, onlooker bees and maximum cycle number (MCN) are set. Next, the job's processing time on each machine and the job's machine sequence will be given at this step. Examples can be found in Tables 1 and 2, respectively.

In our solution representation, a solution in JSSP is an operation scheduling list, which is represented as a food source (x) in our Best-so-far ABC algorithm. Each dimension in a food source represents one operation of a job. Each job appears exactly m times in an operation scheduling list. For the n-job and m-machine problem, each food source contains $n \times m$ dimensions corresponding to $n \times m$ operations. This representation is illustrated in Fig. 3. For Fig. 3, J_i stands for the operation of job i. Since each job has three operations, it occurs three times in the operation scheduling list.

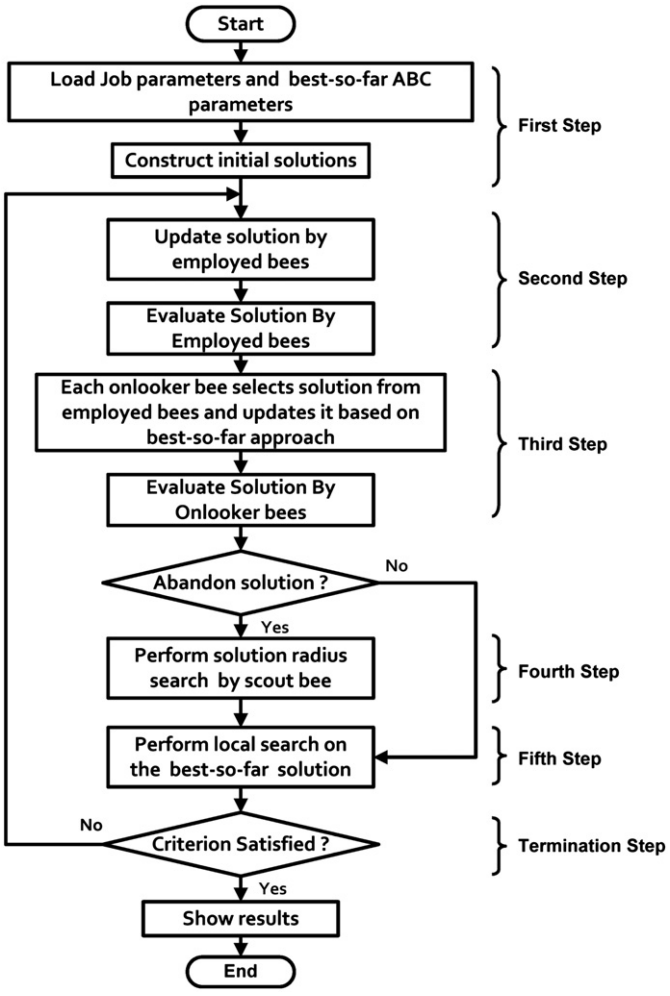

Fig. 2. Best-so-far ABC algorithm flowchart for JSSP.

Table 1
An example of job processing time on each machine for 3-job × 3-machine.

	M_1	M_2	M_3
J_1	15	25	18
J_2	7	10	20
J_3	12	10	8

Table 2
An example of job machine sequence for 3-job × 3-machine.

	O_1	O_2	O_3
J_1	M_1	M_2	M_3
J_2	M_2	M_3	M_1
J_3	M_3	M_1	M_2

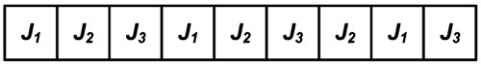

Fig. 3. Example of operation scheduling list representation for 3-job × 3-machine.

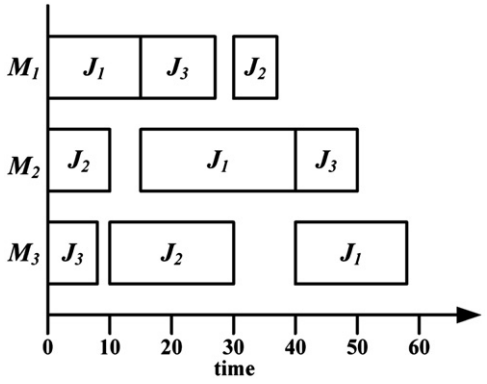

Fig. 4. Feasible schedule constructed from the operation scheduling list in Fig. 3.

The interpretation of the example above is as follows. As we scan the operation scheduling list from left to right, the first J_1 corresponds to the first operation of job J_1, which will be processed on machine M_1, the first J_2 corresponds to the first operation of job J_2, which will be processed on M_2, the first J_3 corresponds to the first operation of job J_3, which will be processed on M_3, the second J_1 corresponds to the second operation of job J_1, which will be processed on machine M_2, the second J_2 corresponds to the second operation of job J_2, which will be processed on M_3, and the second J_3 corresponds to the second operation of job J_3, which will be processed on M_1. Thus, the feasible schedule can be constructed as shown in Fig. 4.

The fitness of each food source $f(x)$ is determined by the inverse of its makespan value ($C_{max}(x)$), which is calculated during the selection of feasible solutions. This mapping is shown in Fig. 5.

The Second Step

We employ the concept of the updating of a candidate food source based on a neighboring food source using the following equation

$$\upsilon_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{4.1}$$

The candidate food sources are updated by employed bees. Position Base Crossover (PBX) (Kellegoz and Toklu, 2008) is the mechanism used to update the employed bees' old food sources (x_{ij}) to new food sources (υ_{ij}) based on their neighboring food sources (x_{kj}) randomly taken from other employed bees. An example of PBX is shown in Fig. 6.

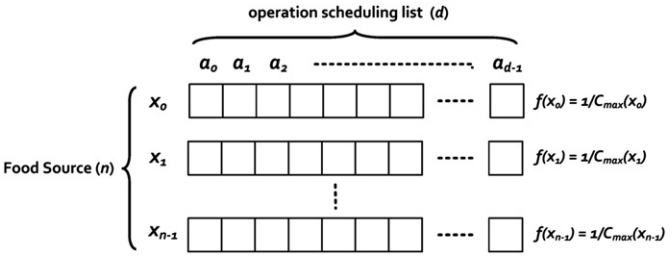

Fig. 5. Mapping between the food sources and operation scheduling list where $f(x_i)=1/C_{\max}(x_i)$.

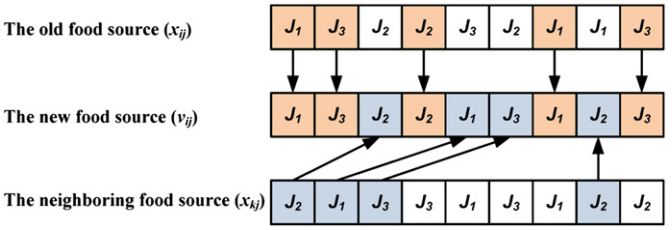

Fig. 6. Exchanging information with a neighboring food source based on PBX method.

Based on the PBX method, a set of job operations from the old food source is selected randomly. Each dimension in the operation scheduling list of the old food source is selected to produce the new position with the probability of 0.5. This probability is empirically chosen by the experimenter to allow the new food source to be generated from both the old food source and the neighboring food source in the same proportion. The jobs already selected from the old food source are ignored and will not be selected again from the neighboring food source. The job operations on the neighboring food source that are not yet selected from the old food source will be selected and placed into empty positions from the left to the right of the operation scheduling list in the new food source. To guarantee that each job will be included exactly m times in the new food source, if any job has already been selected for m times (from either the old or the neighboring food source), it will be skipped and the next job will be considered.

The old food source (x_{ij}) in the employed bee's memory will be replaced by the new candidate food source (υ_{ij}) if the new position has a better fitness value.

The Third Step

The employed bees share new solutions (υ_{ij}) that they have found with the onlooker bees, who then select these solutions based on probability (P_i) calculated from an equation below

$$P_i = \frac{f(\upsilon_i)}{\sum_{n=1}^{SN} f(\upsilon_i)} \tag{4.2}$$

where $f(\upsilon_i)$ is the fitness value of the food source i and SN is the number of food sources.

After onlooker bees have selected the solutions (x_{ij}) from employed bees, then the best solution (x_{bj}) is identified from among those selected solutions. The best solution (x_{bj}) will replace the best-so-far solution (x_{bj} of the previous iteration) if the new best solution is better and its fitness value (f_b) will be used for guiding the direction of the search space as shown in equation below

$$v_{id} = x_{ij} + \Phi f_b(x_{ij} - x_{bj}) \tag{4.3}$$

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

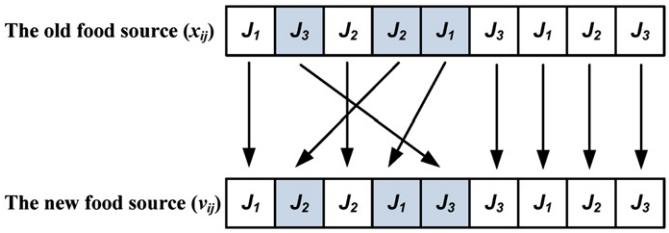

Fig. 7. Updating food source based on radius search by scout bee to find v_{ij}.

The PBX method is also used to update the old solution (x_{ij}) of the onlooker bees to the new solution (v_{ij}) based on the best-so-far food source (x_{bj}) instead of the neighboring food sources (x_{kj}).

The old food source (x_{ij}) in the onlooker bee's memory will be replaced by the new candidate food source (v_{ij}) if the new position has a better fitness value.

The Fourth Step

To avoid suboptimal solutions, the scout bees ignore the old solution and randomly search for new solutions by using the concept of the equation below

$$v_{ij} = x_{ij} + \emptyset_{ij}\left[\omega_{max} - \frac{iteration}{MCN}(\omega_{max} - \omega_{min})\right]x_{ij} \tag{4.4}$$

Based on this concept, the operation scheduling lists (x_{ij}) whose fitness values have not improved after a certain period are abandoned and replaced by new operation scheduling lists (v_{ij}) updated by the scout bees using the radius search method as shown in Fig. 7.

In the radius search method, a set of job operations from the old food source is selected randomly. Each dimension in the operation scheduling list is selected with the probability of ω_{max} in the first iteration and will linearly decrease to ω_{min} in the last iteration. The value of ω_{max} and ω_{min} are fixed to 1 and 0.2, respectively. These parameters were chosen by the experimenter (Banharnsakun et al., 2011). So the first iteration will replace every dimension, that is, will swap the randomly selected solution for the one being abandoned.

In this swapping, the dimensions that are not marked as selected dimension from the old food source will be placed into the same dimension in the new food source. Next, the dimensions marked as selected dimension from the old food source will be placed into the empty positions from the left to right of the operation scheduling list in the new food source.

The Fifth Step

A local search based on the Variable Neighboring Search method (VNS) (Hansen et al., 2008) is performed on the best-so-far solution (x_b of Eq. 4.3) to improve the solution quality. The pseudo code of VNS method is shown in Fig. 8.

Although it seems that VNS would actually find the best solution by itself, it sometimes takes a long time to reach useful solutions whilst solving large scale Job Shop Scheduling. To overcome this shortcoming, our Best-so-far ABC helps VNS to find solutions more quickly by substantially narrowing down the search space.

From Fig. 8, let α and β are the random integer numbers between 1 and nm, *Exchanging_Process*(x',α,β) means exchanging the job operations in solution x' between αth and βth dimensions, $\alpha \neq \beta$. *Inserting_Process*(x',α,β) means removing the job operation in solution x' from the αth dimension and inserting it in the βth dimension.

The example of the exchanging process and the inserting process are shown in Figs. 9 and 10, respectively.

Procedure VNS_Procedure
Get Initial Solution, $x' = x_b$
Set $step = 0$ and $p = 1$
$n = number\ of\ jobs$
$m = number\ of\ machines$
$\alpha = random\_integer\_number[1, nm]$
$\beta = random\_integer\_number[1, nm]\ , \beta \neq \alpha$
$x' = Exchanging\_Process(x', \alpha, \beta)$
$\alpha = random\_integer\_number[1, nm]$
$\beta = random\_integer\_number[1, nm], \beta \neq \alpha$
$x' = Inserting\_Process(x', \alpha, \beta)$
$\alpha = random\_integer\_number[1, nm]$
$\beta = random\_integer\_number[1, nm]\ , \beta \neq \alpha$
$x' = Exchanging\_Process(x', \alpha, \beta)$
While $(step \leq nm * (nm - 1))$
$\alpha = random\_integer\_number[1, nm]$
$\beta = random\_integer\_number[1, nm], \beta \neq \alpha$
If $(p = 1)$ **then** $x'' = Exchanging\_Process(x', \alpha, \beta)$
Else if $(p = 0)$ **then** $x'' = Inserting\_Process(x', \alpha, \beta)$
If $(fitness(x'') \geq fitness(x'))$ then $x' = x''$
Else $p = |p - 1|$
$step = step + 1$
End while
If $(fitness(x') \geq fitness(x_b))$ then $x_b = x'$
End-Procedure

Fig. 8. Variable Neighboring Search (VNS) procedure.

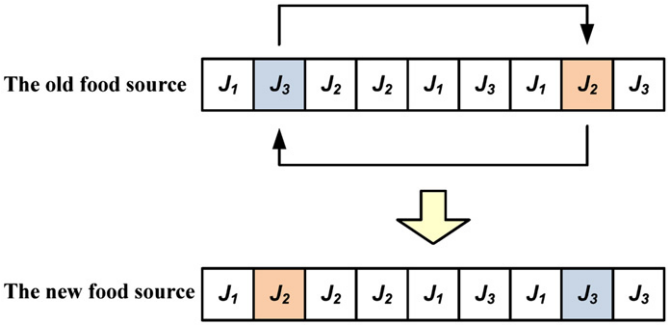

Fig. 9. Exchanging process in VNS method (for new x_b).

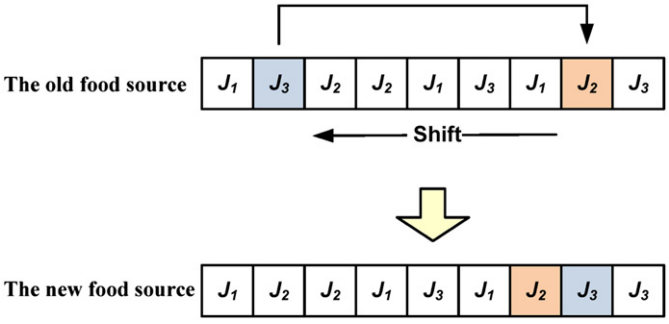

Fig. 10. Inserting process in VNS method (for new x_b).

In Fig. 9, for the exchanging process, suppose the two random numbers α and β generated are 2 and 8, respectively. In this case, the job operations between the 2nd dimension and 8th dimension will be interchanged. In this example, the job operation J_3 at the 2nd dimension and the job operation J_2 at the 8th dimension will be swapped.

In Fig. 10, for the inserting process, suppose the two random numbers α and β generated are 2 and 8, respectively. In this case, the job operation J_3 at the 2nd dimension will be

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

Table 3
Comparison results of the Best-so-far ABC and IABC based on approximately the same number of populations and the same number of iterations.

Instance	Size ($n \times m$)	BKS	Best-so-far ABC		IABC	
			Best	RPE	Best	RPE
ft06	6×6	55	55	0.00	55	0.00
ft10	10×10	930	930	0.00	930	0.00
ft20	20×5	1165	1165	0.00	1165	0.00
la01	10×5	666	666	0.00	666	0.00
la06	15×5	926	926	0.00	926	0.00
la11	20×5	1222	1222	0.00	1222	0.00
la16	10×10	945	**945**	**0.00**	977	3.39
la21	15×10	1046	**1046**	**0.00**	1051	0.48
la26	20×10	1218	1218	0.00	1218	0.00
la31	30×10	1784	1784	0.00	1784	0.00
la36	15×15	1268	**1268**	**0.00**	1275	0.55
Mean				0.00		1.47

removed, the job operations at 3rd dimension through 8th dimension will be shifted to the 2nd dimension, and then insert job operation J_3 in the 8th dimension.

The Termination Step

Steps 2–5 are repeated until the number of iteration reaches the MCN.

5. Experiments

The aim of this experiment is to evaluate the solution quality based on the same parameter settings, i.e. the number of populations and the number of iterations. The Best-so-far algorithm is compared with other approaches including a hybrid Particle Swarm Optimization with Variable Neighboring Search (PSO-VNS) (Tasgetiren et al., 2006), a multiple-type individual enhancement PSO (MPSO) (Lin et al., 2010), a Hybrid Intelligent Algorithm (HIA) (Ge et al., 2008), a Hybrid Evolutionary Algorithm (HEA)

Table 4
Comparison results of the Best-so-far ABC, MPSO, HIA, HEA, and HGA-Param based on approximately the same number of populations and the same number of iterations.

Instance	Size ($n \times m$)	BKS	Best-so-far ABC		MPSO		HIA		HEA		HGA-Param	
			Best	RPE	Best	RPE	Best	RPE	Best	RPE	Best	RPE
ft06	6×6	55	55	0.00	55	0.00	55	0.00	55	0.00	55	0.00
ft10	10×10	930	930	0.00	930	0.00	930	0.00	930	0.00	930	0.00
ft20	20×5	1165	**1165**	0.00	**1165**	0.00	**1165**	0.00	1169	0.34	**1165**	0.00
la01	10×5	666	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00
la02	10×5	655	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00
la03	10×5	597	597	0.00	597	0.00	597	0.00	597	0.00	597	0.00
la04	10×5	590	590	0.00	590	0.00	590	0.00	590	0.00	590	0.00
la05	10×5	593	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00
la06	15×5	926	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00
la07	15×5	890	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00
la08	15×5	863	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00
la09	15×5	951	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00
la10	15×5	958	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00
la11	20×5	1222	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00
la12	20×5	1039	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00
la13	20×5	1150	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00
la14	20×5	1292	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00
la15	20×5	1207	1207	0.00	1207	0.00	1207	0.00	1207	0.00	1207	0.00
la16	10×10	945	945	0.00	945	0.00	945	0.00	945	0.00	945	0.00
la17	10×10	784	784	0.00	784	0.00	784	0.00	784	0.00	784	0.00
la18	10×10	848	848	0.00	848	0.00	848	0.00	848	0.00	848	0.00
la19	10×10	842	842	0.00	842	0.00	842	0.00	–	–	842	0.00
la20	10×10	902	902	0.00	902	0.00	902	0.00	–	–	907	0.55
la21	15×10	1046	1046	0.00	1046	0.00	1046	0.00	1046	0.00	1046	0.00
la22	15×10	927	927	0.00	932	0.54	932	0.54	935	0.86	935	0.86
la23	15×10	1032	1032	0.00	1032	0.00	1032	0.00	1032	0.00	1032	0.00
la24	15×10	935	**935**	0.00	941	0.64	950	1.60	–	–	953	1.93
la25	15×10	977	**977**	0.00	**977**	0.00	979	0.20	–	–	986	0.92
la26	20×10	1218	1218	0.00	1218	0.00	1218	0.00	1218	0.00	1218	0.00
la27	20×10	1235	**1235**	0.00	1239	0.32	1256	1.70	1272	3.00	1256	1.70
la28	20×10	1216	**1216**	0.00	**1216**	0.00	1227	0.90	1227	0.90	1232	1.32
la29	20×10	1152	**1164**	1.04	1173	1.82	1184	2.78	1192	3.47	1196	3.82
la30	20×10	1355	1355	0.00	1355	0.00	1355	0.00	1355	0.00	1355	0.00
la31	30×10	1784	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00
la32	30×10	1850	1850	0.00	1850	0.00	1850	0.00	1850	0.00	1850	0.00
la33	30×10	1719	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00
la34	30×10	1721	1721	0.00	1721	0.00	1721	0.00	1721	0.00	1721	0.00
la35	30×10	1888	1888	0.00	1888	0.00	1888	0.00	1888	0.00	1888	0.00
la36	15×15	1268	**1268**	0.00	1278	0.79	1281	1.03	1287	1.50	1279	0.87
la37	15×15	1397	**1397**	0.00	1411	1.00	1415	1.29	1415	1.29	1408	0.79
la38	15×15	1196	**1196**	0.00	1208	1.00	1213	1.42	1213	1.42	1219	1.92
la39	15×15	1233	**1233**	0.00	**1233**	0.00	1246	1.05	1245	0.97	1246	1.05
la40	15×15	1222	**1224**	0.16	1225	0.25	1240	1.47	1242	1.64	1241	1.55
Mean				0.03		0.15		0.33		0.39		0.44

"–" stands for "not available".

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

(Ge et al., 2007), a Hybrid Genetic Algorithm using parameterized active schedules (HGA-Param) (Goncalves et al., 2005), and an Improved Artificial Bee Colony algorithm (IABC) (Yao et al., 2010). The performance of our Best-so-far ABC algorithm is evaluated by testing them on the following 62 benchmark problems taken from the Operations Research Library (OR-Library) (Beasley, 1990).

- 3 problems from Fisher and Thompson (1963): referred as *ft06*, *ft10*, and *ft20*.
- 40 problems from Lawrence (1984): referred as *la01–la40*.
- 5 problems from Adams et al. (1988): referred as *abz05–abz09*.
- 10 problems from Applegate and Cook (1991): referred as *orb01–orb10*.
- 4 problems from Yamada and Nakano (1992): referred as *yn01–yn04*.

The size of these problem instances range from 6 to 30 jobs and 5 to 20 machines.

The objective is to find the minimum makespan value from these benchmark problems. The numbers of employed and onlooker bees were set to 25. The MCN was set to 200. Each of the experiments was repeated 20 times with different random seeds. Our algorithm was coded in C++ and ran on a PC with a 2.83-GHz Intel Core 2 Quad CPU and 4 GB of memory. We used the results reported in Goncalves et al. (2005), Tasgetiren et al. (2006), Lin et al. (2010), Ge et al. (2008), Ge et al. (2007), and Yao et al. (2010) for the comparison. As a result, some problem instances cannot be evaluated in all aforementioned algorithms.

6. Results and discussion

We compare our results with the results in other aforementioned studies in Tables 3–5. In the Tables , "instance" means the problem name, "size" means the problem size of n jobs on m machines, "BKS" means the best known solution for the instance reported by Jain and Meeran (1999), "Best" means the best solution found by each algorithm, "Average" and "S.D." means the average and standard deviation, respectively, of the results over 20 runs, and "RPE" means the relative percent error with respect to the best known solution, which is calculated from the equation below

$$\mathrm{RPE} = \frac{(\mathrm{Best} - \mathrm{BKS})}{\mathrm{BKS}} \times 100 \tag{6.1}$$

In each table of comparison results, boldface represents problems in which our Best-so-far ABC reached a higher quality solution than at least one of the previously reported algorithms. It can be noticed from the "Best", "Average", "S.D.", and "RPE" values in Tables 3–5.

The first comparison is based on the results of an Improved Artificial Bee Colony algorithm (IABC) proposed by Yao et al. (2010). The comparison results are shown in Table 3.

From Table 3, it can be seen that the Best-so-far ABC generates better results in term of the best makespan (Best) and relative percent error (RPE) than the IABC. The Best-so-far ABC method is able to find all of the best known solutions for the 11 instances reported and average of RPE for Best-so-far ABC is 0% whereas the

Table 5
Comparison results of the Best-so-far ABC and PSO-VNS based on approximately the same number of populations and the same number of iterations.

Instance	Size ($n \times m$)	BKS	Best-so-far ABC				PSO-VNS			
			Best	Average	S.D.	RPE	Best	Average	S.D.	RPE
abz05	10 × 10	1234	1234	1234.75	1.48	0.00	1234	1236.25	2.31	0.00
abz06	10 × 10	943	943	943.00	0.00	0.00	943	943.00	0.00	0.00
abz07	20 × 15	656	661	669.60	4.31	0.76	**659**	670.10	5.75	0.46
abz08	20 × 15	665	674	678.45	3.58	1.35	674	682.30	5.52	1.35
abz09	20 × 15	679	684	693.85	5.12	0.74	688	697.55	5.79	1.33
ft10	10 × 10	930	930	931.45	2.98	0.00	930	938.45	9.71	0.00
ft20	20 × 5	1165	1165	1169.45	5.81	0.00	1165	1175.25	5.30	0.00
la16	10 × 10	945	945	945.35	0.49	0.00	945	948.50	9.08	0.00
la19	10 × 10	842	842	842.00	0.00	0.00	842	844.00	3.78	0.00
la21	15 × 10	1046	**1046**	1050.00	3.23	0.00	1047	1053.80	6.01	0.10
la22	15 × 10	927	927	928.35	2.37	0.00	927	930.55	2.95	0.00
la24	15 × 10	935	935	939.10	1.89	0.00	935	939.70	3.63	0.00
la25	15 × 10	977	977	981.40	2.87	0.00	977	981.45	4.74	0.00
la27	20 × 10	1235	1235	1245.10	5.98	0.00	1235	1248.10	10.09	0.00
la28	20 × 10	1216	1216	1216.10	0.45	0.00	1216	1216.25	0.64	0.00
la29	20 × 10	1152	1164	1173.80	5.63	1.04	1164	1176.70	10.55	1.04
la36	15 × 15	1268	1268	1275.05	3.95	0.00	1268	1279.30	6.98	0.00
la37	15 × 15	1397	1397	1409.15	7.53	0.00	1397	1410.90	7.74	0.00
la38	15 × 15	1196	1196	1204.95	4.35	0.00	1196	1212.50	14.96	0.00
la39	15 × 15	1233	1233	1237.95	3.73	0.00	1233	1240.00	4.10	0.00
la40	15 × 15	1222	1224	1226.25	2.53	0.16	1224	1227.60	3.80	0.16
orb01	10 × 10	1059	1059	1073.00	8.96	0.00	1059	1076.05	10.58	0.00
orb02	10 × 10	888	**888**	889.05	0.39	0.00	889	889.45	1.79	0.11
orb03	10 × 10	1005	1005	1018.30	8.57	0.00	1005	1034.55	22.95	0.00
orb04	10 × 10	1005	1005	1010.90	3.75	0.00	1005	1011.30	6.42	0.00
orb05	10 × 10	887	887	890.10	2.00	0.00	887	892.45	4.81	0.00
orb06	10 × 10	1010	**1010**	1015.65	5.06	0.00	1013	1018.80	5.73	0.30
orb07	10 × 10	397	397	397.60	1.85	0.00	397	398.60	2.56	0.00
orb08	10 × 10	899	899	906.85	6.29	0.00	899	913.40	14.19	0.00
orb09	10 × 10	934	934	938.45	3.73	0.00	934	939.45	4.27	0.00
orb10	10 × 10	944	944	944.00	0.00	0.00	944	944.00	0.00	0.00
yn01	20 × 20	888	**891**	897.60	3.25	0.34	893	901.15	6.56	0.56
yn02	20 × 20	909	911	919.90	5.50	0.22	**910**	925.85	6.43	0.11
yn03	20 × 20	893	**897**	904.05	3.46	0.45	902	908.40	5.23	1.01
yn04	20 × 20	968	**972**	984.05	4.21	0.41	973	987.05	8.87	0.52
Mean				993.85	3.60	0.16		996.94	6.40	0.20

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

IABC is able to find only 8 best known solutions among 11 instances reported with an average RPE of 1.47%.

The second comparison is based on the results of a multiple-type individual enhancement PSO (MPSO) proposed by Lin et al. (2010), a Hybrid Intelligent Algorithm (HIA) proposed by Ge et al. (2008), a Hybrid Evolutionary Algorithm (HEA) proposed by Ge et al. (2007), and a Hybrid Genetic Algorithm using parameterized active schedules (HGA-Param) proposed by Goncalves et al. (2005). The comparison results are shown in Table 4.

According to Table 4, for instances *ft06*, *ft10*, *ft20*, *and la01–la40*, our Best-so-far ABC can find the best known solution (Best) for 41 of 43 instances. These results are better than MPSO, HIA, HEA, and HGA, which can only find 35, 32, 29, and 31 best know solutions among 43 instances, respectively. Moreover, in term of the average RPE, the Best-so-far ABC generates better results than these aforementioned approaches. The average RPE for the Best-so-far ABC is 0.03% whereas MPSO, HIA, HEA, and HGA got 0.15%, 0.33%, 0.39%, and 0.44%, respectively.

The last comparison is based on the results of hybrid Particle Swarm Optimization with Variable Neighboring Search (PSO-VNS) by Tasgetiren et al. (2006). The comparison results are shown in Table 5.

As shown in Table 5, the Best-so-far ABC can find the best known solutions (Best) for 26 of 35 instances reported where the PSO-VNS only finds the best know solutions with 23 of 35 instances reported. Moreover, the average RPE in the 35 instances reported for PSO-VNS is 0.20% whereas the average RPE for Best-so-far ABC is 0.16%. This shows that the Best-so-far ABC does better in finding the best known solution than the PSO-VNS. The mean of the average of makespan for the Best-so-far ABC is 993.85 whereas PSO-VNS is 996.94. The results indicate that the Best-so-far ABC gives slightly better makespan values on average than the PSO-VNS. It is also clear that the Best-so-far algorithm is more robust than the PSO-VNS algorithm as shown by the lower standard deviations in the S.D. column. The average of the standard deviations for the Best-so-far ABC is only 3.60 whereas it is 6.40 for the PSO-VNS. Note that we do not report standard deviations for other comparisons because standard deviations are not reported in the papers presenting the other algorithms.

In addition, the average RPE for the Best-so-far ABC on all 62 benchmark problems is 0.09%. A comparison of all aforementioned approaches is provided in Appendix A.

7. Conclusions

In this paper, a Best-so-far ABC algorithm for solving the Job Shop Scheduling Problem is proposed. We have replaced the neighboring solutions based approach with the Best-so-far technique in order to increase the local search ability of the onlooker bees. This will bias the solution selected by onlooker bees towards the optimal solution more quickly. The use of set theory was introduced to describe the mapping between our Best-so-far ABC method and the combinatorial optimization problem.

The experiment was benchmarked against six state-of-the-art methods from three well known algorithms. There are three methods from Particle Swamp Optimization, including hybrid Particle Swarm Optimization with Variable Neighboring Search (PSO-VNS), multiple-type individual enhancement PSO (MPSO), and Hybrid Intelligent Algorithm (HIA). For Genetic Algorithms, Hybrid Evolutionary Algorithm (HEA) and Hybrid Genetic Algorithm using parameterized active schedules (HGA-Param) was included. Finally, Improved Artificial Bee Colony algorithm (IABC) from ABC algorithm was used.

The results obtained from our proposed method show that Best-so-far ABC can find the best known solution more effectively than other aforementioned approaches. This difference is statistically significant for three of the six alternative algorithms used for comparison. Moreover, the Best-so-far ABC algorithm gives a better average makespan value. The results also show that our Best-so-far ABC is more robust than the PSO-VNS method as shown by the value of SD in Table 5.

Thus, we can conclude that our Best-so-far ABC is effective from both the perspective of solution quality and algorithm robustness. The algorithm can serve as an alternative method in the Job Shop Scheduling domain.

In the future, we will investigate JSSP in a case where "job interrupt" is permitted. Our proposed algorithm may split a job into a number of smaller sub-jobs. This will allow each job to be preempted by other jobs with higher priorities. The preempted job can later be resumed when a machine is free. With this variation, we believe that Best-so-far ABC will be even more practical for use in most job shop scheduling problems.

Acknowledgments

This work is supported by the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant no. PHD/0038/2552.

Appendix A

See Appendix Table A1 below.

Table A1
Comparison of results between the Best-so-far ABC and other approaches.

Instance	Size ($n \times m$)	BKS	Best-so-far ABC				PSO-VNS				MPSO	HIA	HEA	HGA-Param	IABC
			Best	Average	S.D.	RPE	Best	Average	S.D.	RPE	Best	Best	Best	Best	Best
abz05	10×10	1234	1234	1234.75	1.48	0.00	1234	1236.25	2.31	0.00	–	–	–	–	–
abz06	10×10	943	943	943.00	0.00	0.00	943	943.00	0.00	0.00	–	–	–	–	–
abz07	20×15	656	661	669.60	4.31	0.76	**659**	670.10	5.75	0.46	–	–	–	–	–
abz08	20×15	665	674	678.45	3.58	1.35	674	682.30	5.52	1.35	–	–	–	–	–
abz09	20×15	679	**684**	693.85	5.12	0.74	688	697.55	5.79	1.33	–	–	–	–	–
ft06	6×6	55	55	55.00	0.00	0.00	–	–	–	–	55	55	55	55	55
ft10	10×10	930	930	931.45	2.98	0.00	930	938.45	9.71	0.00	930	930	930	930	930
ft20	20×5	1165	**1165**	1169.45	5.81	0.00	**1165**	1175.25	5.30	0.00	**1165**	**1165**	1169	**1165**	**1165**
la01	10×5	666	666	666.00	0.00	0.00	–	–	–	–	666	666	666	666	666
la02	10×5	655	655	655.00	0.00	0.00	–	–	–	–	655	655	655	655	–
la03	10×5	597	597	597.00	0.00	0.00	–	–	–	–	597	597	597	597	–
la04	10×5	590	590	590.00	0.00	0.00	–	–	–	–	590	590	590	590	–
la05	10×5	593	593	593.00	0.00	0.00	–	–	–	–	593	593	593	593	–

Table A1 (*continued*)

Instance	Size ($n \times m$)	BKS	Best-so-far ABC				PSO-VNS				MPSO	HIA	HEA	HGA-Param	IABC
			Best	Average	S.D.	RPE	Best	Average	S.D.	RPE	Best	Best	Best	Best	Best
la06	15 × 5	926	926	926.00	0.00	0.00	–	–	–	–	926	926	926	926	926
la07	15 × 5	890	890	890.00	0.00	0.00	–	–	–	–	890	890	890	890	–
la08	15 × 5	863	863	863.00	0.00	0.00	–	–	–	–	863	863	863	863	–
la09	15 × 5	951	951	951.00	0.00	0.00	–	–	–	–	951	951	951	951	–
la10	15 × 5	958	958	958.00	0.00	0.00	–	–	–	–	958	958	958	958	–
la11	20 × 5	1222	1222	1222.00	0.00	0.00	–	–	–	–	1222	1222	1222	1222	1222
la12	20 × 5	1039	1039	1039.00	0.00	0.00	–	–	–	–	1039	1039	1039	1039	–
la13	20 × 5	1150	1150	1150.00	0.00	0.00	–	–	–	–	1150	1150	1150	1150	–
la14	20 × 5	1292	1292	1292.00	0.00	0.00	–	–	–	–	1292	1292	1292	1292	–
la15	20 × 5	1207	1207	1207.00	0.00	0.00	–	–	–	–	1207	1207	1207	1207	–
la16	10 × 10	945	945	945.35	0.49	0.00	945	948.50	9.08	0.00	945	945	945	945	977
la17	10 × 10	784	784	784.00	0.00	0.00	–	–	–	–	784	784	784	784	–
la18	10 × 10	848	848	848.00	0.00	0.00	–	–	–	–	848	848	848	848	–
la19	10 × 10	842	842	842.00	0.00	0.00	842	844.00	3.78	0.00	842	842	–	842	–
la20	10 × 10	902	902	905.25	2.45	0.00	–	–	–	–	902	902	–	907	–
la21	15 × 10	1046	**1046**	1050.00	3.23	0.00	1047	1053.80	6.01	0.10	**1046**	**1046**	**1046**	**1046**	1051
la22	15 × 10	927	927	928.35	2.37	0.00	927	930.55	2.95	0.00	932	932	935	935	–
la23	15 × 10	1032	1032	1032.00	0.00	0.00	–	–	–	–	1032	1032	1032	1032	–
la24	15 × 10	935	**935**	939.10	1.89	0.00	**935**	939.70	3.63	0.00	941	950	–	953	–
la25	15 × 10	977	**977**	981.40	2.87	0.00	**977**	981.45	4.74	0.00	**977**	979	–	986	–
la26	20 × 10	1218	1218	1218.00	0.00	0.00	–	–	–	–	1218	1218	1218	1218	1218
la27	20 × 10	1235	**1235**	1245.10	5.98	0.00	**1235**	1248.10	10.09	0.00	1239	1256	1272	1256	–
la28	20 × 10	1216	**1216**	1216.10	0.45	0.00	**1216**	1216.25	0.64	0.00	**1216**	1227	1227	1232	–
la29	20 × 10	1152	**1164**	1173.80	5.63	1.04	**1164**	1176.70	10.55	1.04	1173	1184	1192	1196	–
la30	20 × 10	1355	1355	1355.00	0.00	0.00	–	–	–	–	1355	1355	1355	1355	–
la31	30 × 10	1784	1784	1784.00	0.00	0.00	–	–	–	–	1784	1784	1784	1784	1784
la32	30 × 10	1850	1850	1850.00	0.00	0.00	–	–	–	–	1850	1850	1850	1850	–
la33	30 × 10	1719	1719	1719.00	0.00	0.00	–	–	–	–	1719	1719	1719	1719	–
la34	30 × 10	1721	1721	1721.00	0.00	0.00	–	–	–	–	1721	1721	1721	1721	–
la35	30 × 10	1888	1888	1888.00	0.00	0.00	–	–	–	–	1888	1888	1888	1888	–
la36	15 × 15	1268	**1268**	1275.05	3.95	0.00	**1268**	1279.30	6.98	0.00	1278	1281	1287	1279	1275
la37	15 × 15	1397	**1397**	1409.15	7.53	0.00	**1397**	1410.90	7.74	0.00	1411	1415	1415	1408	–
la38	15 × 15	1196	**1196**	1204.95	4.35	0.00	**1196**	1212.50	14.96	0.00	1208	1213	1213	1219	–
la39	15 × 15	1233	**1233**	1237.95	3.73	0.00	**1233**	1240.00	4.10	0.00	**1233**	1246	1245	1246	–
la40	15 × 15	1222	**1224**	1226.25	2.53	0.16	**1224**	1227.60	3.80	0.16	1225	1240	1242	1241	–
orb01	10 × 10	1059	1059	1073.00	8.96	0.00	1059	1076.05	10.58	0.00	–	–	–	–	–
orb02	10 × 10	888	**888**	889.05	0.39	0.00	889	889.45	1.79	0.11	–	–	–	–	–
orb03	10 × 10	1005	1005	1018.30	8.57	0.00	1005	1034.55	22.95	0.00	–	–	–	–	–
orb04	10 × 10	1005	1005	1010.90	3.75	0.00	1005	1011.30	6.42	0.00	–	–	–	–	–
orb05	10 × 10	887	887	890.10	2.00	0.00	887	892.45	4.81	0.00	–	–	–	–	–
orb06	10 × 10	1010	**1010**	1015.65	5.06	0.00	1013	1018.80	5.73	0.30	–	–	–	–	–
orb07	10 × 10	397	397	397.60	1.85	0.00	397	398.60	2.56	0.00	–	–	–	–	–
orb08	10 × 10	899	899	906.85	6.29	0.00	899	913.40	14.19	0.00	–	–	–	–	–
orb09	10 × 10	934	934	938.45	3.73	0.00	934	939.45	4.27	0.00	–	–	–	–	–
orb10	10 × 10	944	944	944.00	0.00	0.00	944	944.00	0.00	0.00	–	–	–	–	–
yn01	20 × 20	888	**891**	897.60	3.25	0.34	893	901.15	6.56	0.56	–	–	–	–	–
yn02	20 × 20	909	911	919.90	5.50	0.22	**910**	925.85	6.43	0.11	–	–	–	–	–
yn03	20 × 20	893	**897**	904.05	3.46	0.45	902	908.40	5.23	1.01	–	–	–	–	–
yn04	20 × 20	968	**972**	984.05	4.21	0.41	973	987.05	8.87	0.52	–	–	–	–	–

"–" stands for "not available".

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for jobshop scheduling. Manage. Sci. 34, 391–401.

Applegate, D., Cook, W., 1991. A computational study of jobshop problem. ORSA J. Comput. 3, 149–156.

Asadzadeh, L., Zamanifar, K., 2010. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. Math. Comput. Model. 52, 1957–1965.

Banharnsakun, A., Achalakul, T., Sirinaovakul, B., 2011. The best-so-far selection in Artificial Bee Colony Algorithm. Appl. Soft Comput. 11, 2888–2901.

Beasley, J.E., 1990. Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. 14, 1069–1072.

Biesmeijer, J.C., Seeley, T.D., 2005. The use of waggle dance information by honey bees throughout their foraging careers. Behav. Ecol. Sociobiol. 59, 133–142.

Carlier, J., Pinson, E., 1989. An algorithms for solving the job-shop problem. Manage. Sci. 35, 164–176.

Chong, C.S., Low, M.Y.H., 2006. A bee colony optimization algorithm to job shop scheduling. In: Proceedings of the 2006 Winter Simulation Conference, pp. 1954–1961.

Fisher, H., Thompson, G.L., 1963. Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (Eds.), Industrial SchedulingPrentice-Hall, Englewood Cliffs, pp. 225–251.

Fisher, M., 1981. The lagrangian relaxation method for solving integer programming problems. Manage. Sci. 27, 1–18.

French, S., 1982. Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop. Wiley, New York, USA.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Math. Oper. Res. 1, 117–129.

Ge, H., Du, W., Qian, F., 2007. A hybrid algorithm based on particle swarm optimization and simulated annealing for job shop scheduling. In: Proceedings of the Third International Conference on Natural Computation, vol. 3, pp. 715–719.

Ge, H.-W., Sun, L., Liang, Y.-C., Qian, F., 2008. An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling. IEEE Trans. Syst., Man Cybern.—Part A: Syst. Hum. 38, 358–368.

Goncalves, J.F., Mendes, J.J.D.M., Resende, M.G.C., 2005. A hybrid genetic algorithm for the job shop scheduling problem. Eur. J. Oper. Res. 167, 77–95.

Hansen, P., Mladenović, N., Pérez, J.A.M., 2008. Variable neighbourhood search: methods and applications. 4OR: A Q. J. Oper. Res. 6, 319–360.

Heinonen, J., Pettersson, F., 2007. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. Appl. Math. Comput. 187, 989–998.

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003

Jain, A.S., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. Eur. J. Oper. Res. 113, 390–434.
Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06. Engineering Faculty, Computer Engineering Department, Erciyes University, Turkey.
Karaboga, D., Akay, B., 2009. A survey: algorithms simulating bee swarm intelligence. Artif. Intell. Rev. 31, 61–85.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39, 459–471.
Kellegoz, T., Toklu, B., 2008. Comparing efficiencies of genetic crossover operations for one machine total weighted tardiness problem. Appl. Math. Comput. 199, 590–598.
Lawrence, S., 1984. Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques. Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.
Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., Lai, J.-L., Kuo, I.-H., 2010. An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Syst. Appl. 37, 2629–2636.
Pan, Q.-K., Tasgetiren, M.F., Suganthan, P.N., Chua, T.J., 2011. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inf. Sci. 181, 2455–2468.
Rao, R.S., Narasimham, S.V.L., Ramalingaraju, M., 2008. Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. Int. J. Electr. Power Energy Syst. Eng. 1, 708–714.
Sabat, S.L., Udgata, S.K., Abraham, A., 2010. Artificial bee colony algorithm for small signal model parameter extraction of MESFET. Eng. Appl. Artif. Intell. 23, 689–694.
Singh, A., 2009. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl. Soft Comput. 9, 625–631.
Tasgetiren, M.F., Sevkli, M., Liang, Y.-C., Yenisey, M.M., 2006. A Particle Swarm Optimization and Differential Evolution Algorithms for Job Shop Scheduling Problem. Int. J. Oper. Res. 3, 120–135.
Watanabe, M., Ida, K., Gen, M., 2005. A genetic algorithm with modified crossover operator and search area adaptation for the Job-Shop Scheduling Problem. Comput. Ind. Eng. 48, 743–752.
Yamada, T., Nakano, R., 1992. A genetic algorithm applicable to large scale job shop problems, In: Manner, R., Manderick, B., (Eds.), Proceedings of the Second International Workshop on Parallel Problem Solving from Nature, vol. 2. Elsevier, Amsterdam, pp. 281–290.
Yang, X.S., 2008. Introduction to Computational Mathematics. World Scientific Publishing, New Jersey, USA.
Yao, B., Yang, C., Hu, J., Yin, G., Yu, B., 2010. An improved artificial bee colony algorithm for job shop problem. Appl. Mech. Mater. 26–28, 657–660.

Please cite this article as: Banharnsakun, A., et al., Job Shop Scheduling with the Best-so-far ABC. Engineering Applications of Artificial Intelligence (2011), doi:10.1016/j.engappai.2011.08.003