Applied Mathematics and Computation 247 (2014) 266–280

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Biogeography-based optimization for optimal job scheduling in cloud computing

Sung-Soo Kim [a], Ji-Hwan Byeon [b], Hong Yu [c], Hongbo Liu [d,e,*]

[a] Department of Industrial Engineering, Kangwon National University, Chunchon 200-701, Republic of Korea
[b] Kaiem Co., LTD., Seoul 152-780, Republic of Korea
[c] School of Information Engineering, Dalian Ocean University, Dalian 116023, China
[d] School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
[e] Institute for Neural Computation, University of California San Diego, La Jolla, CA 92093, USA

ARTICLE INFO

Keywords:
Biogeography-based optimization (BBO)
Swarm intelligence
Job scheduling
Cloud computing

ABSTRACT

In cloud computing, the resources are dynamic and their performance or load can change frequently over time. Cloud resource management needs the functionality for NP-complete scheduling of jobs. The objective of this paper is to optimize the job scheduling using biogeography-based optimization (BBO). BBO migration is used to change existing solutions and to adapt new good solutions. BBO offers the advantage of adaptive process, which is developed for binary integer job scheduling problem in cloud computing. Experimental results show that the performance of the proposed methods are better than the considered other methods in job scheduling problems.

1. Introduction

Cloud computing is a large-scale distributed computing paradigm. It is the evolution of the traditional distributed computing and grid computing [1,2]. The computing resources available in cloud are highly dynamic and possibly heterogeneous. It differs from traditional distributed computing [3]. The computational resources are geographically distributed and are shared among jobs in cloud. Job scheduling is one of the important activities performed in grid and cloud computing environments, as its objective is to deliver available computing resources as services over networks [4,5]. Job scheduling is known to be NP-complete [6]. It is drawing researcher's attention worldwide because of its practical importance and its complexity [7,8]. Unfortunately, scheduling algorithms in traditional distributed computing cannot work well in cloud computing as its large-scale and dynamic.

Swarm intelligence is an innovative distributed intelligent paradigm whereby the intelligence that emerges in nature from the organic evolution and the collective behaviors of unsophisticated individuals interacting locally with their environment is exploited to develop optimization algorithms that identify optimum solutions efficiently in complex search spaces [9,10]. Within these paradigm algorithms have been developed that mimic organic evolution process or swarming behaviors observed in swarms of bees, colonies of ants, flocks of birds, and even human social behavior, from which intelligence is seen to emerge [11–13]. The population based algorithm can be classified into two major categories. The first is evolutionary algorithms, such as genetic algorithm (GA), simulated annealing (SA). The second is swarm intelligent based algorithms, such as

* Corresponding author at: Institute for Neural Computation, University of California San Diego, La Jolla, CA 92093, USA.
E-mail addresses: kimss@kangwon.ac.kr (S.-S. Kim), benjy86@nate.com (J.-H. Byeon), yuhong@dlou.edu.cn (H. Yu), hongbo@sccn.ucsd.edu (H. Liu).

http://dx.doi.org/10.1016/j.amc.2014.09.008
0096-3003/

artificial bee colony (ABC) and particle swarm optimization (PSO). These algorithms have been presented as a means of efficiently scheduling job in traditional distributed computing and grid computing [14,8,15].

Recently, Simon has proposed a novel biology based optimization method called biogeography-based optimization (BBO) and compared it with other population based optimization methods [16]. Simon has also mentioned that BBO has advantages for high-dimension problems [17]. We will discuss the related works and methodologies used in earlier studies in Section 2. Motivation of our study is to present BBO algorithms for job scheduling in cloud computing. First, we introduce the standard BBO for job scheduling as BBO1, which generates the valid solutions for initializing population. The BBO2 updates the solutions if new generated solution is better than old ones after migrating for converged search and mutating for diversified search.

The rest of the paper is organized as follows. Related works about job scheduling in distributed computing, grid computing and cloud computing environment is provided briefly in Section 2. We discuss the related theoretical foundations in Section 3. In Section 4, our proposed BBOs are described in detail. Experiment results and analysis are presented in Section 5 and some conclusions are provided towards the end.

2. Related work

Job scheduling is mapping a set of job to a set of resources to effectively and efficiently utilize the computing capabilities and storage capabilities of parallel system, large-scale cluster system, distributed system, grid and cloud. In the past decades, many researchers focus on job scheduling and a multitude of different algorithms have been proposed to solve this problem [18–20]. Due to the NP-complete of the problem, no feasible exact solutions are proposed. Therefore the approximation algorithms that are intent to find a near optimal solution are popular. Heuristic algorithms are suitable approach for solving NP-complete problem [21,22]. The most popular heuristic algorithms are genetic algorithm (GA) [23], simulated annealing (SA) [24], Particle swarm optimization (PSO) [25] and Artificial Bee Colony Optimization (ABC) [26].

Genetic algorithms based job scheduling approaches in distributed environments were studied by Hou [27] and Wang [28]. Hou [27] proposed a genetic algorithm for deterministic model based multiprocessor scheduling problem, in which the execution time and the relationship between jobs are known. Wang et al. [28] proposed a genetic algorithm based approach for job scheduling in heterogeneous computing environments, in which the tasks are decomposed into subtasks that have data dependencies. The proposed algorithm is effective to the static job scheduling. Liu et al. [14] proposed fuzzy particle swarm optimization and verified the performance of PSO compared with GA and SA. The approach to scheduling jobs in computational clouds is based on using fuzzy matrices to represent the position and velocity of the particles in PSO for mapping the job scheduling and the particle. The object function of the scheduling model is to complete the tasks in a minimum period of time as well as utilizing the resources. The performance of the fuzzy swarm optimization is better than that of GA and SA. Kim et al. [8] introduced an efficient binary artificial bee colony algorithm as an enhancement of binary artificial bee colony algorithm for solving the makespan minimization problem in grid computing job scheduling. They demonstrate theoretically that the algorithm converges with a probability of 1 towards the global optimal. However, the job scheduling in cloud is highly dynamic and massively scalable. The job scheduling algorithms in grid and traditional distributed systems cannot work well in cloud. It is necessary to explore effective and efficient algorithm for job scheduling in cloud.

Recently, Simon [16] provided a general presentation of the new optimization method called biogeography-based optimization (BBO). He compares and contrasts BBO with other population-based optimization methods. He also applies BBO to the real-world problem of sensor selection for aircraft engine health estimation. Simon [29] develops a Markov analysis of BBO, including the option of elitism. They also show that elitism is not necessary for all problems, but for some problems it can significantly improve performance. They use elitism in order to retain the best solutions in the population from one generation to the next. Simon [17] illustrated that BBO is a generalization of a GA/GUR (genetic algorithm with global uniform recombination) when immigration rate is 1. The authors of this paper mention that BBO outperforms genetic algorithm with global uniform recombination for all problem sizes and all population sizes. They also mention that BBO has advantages for high-dimension problems and with large populations and BBO performs significantly better than both GA with single-point crossover and GA/GUR for lower mutation rates. They found that BBO consistently performs much better on traveling salesman problem benchmarks, usually performs better on graph coloring benchmarks, and sometimes performs better on bin packing benchmarks. Ma [30] proposed the blended BBO that generally outperforms standard BBO on a set of benchmark problems. The offspring are obtained by combining parents' genes instead of copying a parent's gene to a child chromosome in blended crossover. A new solution feature in a BBO solution is comprised of two components that are the migration of features from another solution and itself. Roy [31] proposed BBO for solving constrained optimal power flow problems in power systems. In this paper, we present BBO algorithms for job scheduling in cloud computing effectively.

3. Problem definition

Cloud computing is a novel computation model, in which the resources are shared by using virtualization technologies over internet [32]. It makes required resources of job manifest in the form of a virtual machine. The open source

virtualization software such as Xen [33] is used for the Cloud providing virtualized computing infrastructure so that multiple applications can share the same resources and each application is running within a cloud node that is a set of computational resources with limited capacities. Every application is completely different and is independent and has no link between each other. The cloud nodes are also independences. Each cloud node is scheduled as an independence unit.

The general framework of cloud is shown in Fig. 1. The users submit their jobs to the job broker who maintains the metadata of jobs, such as the processing requirements. The cloud nodes manager monitors the cloud nodes, at the same time, the cloud nodes manager maintains the states of the cloud nodes and the attributes of cloud nodes, such as the calculating speed. The job manager maintains the states of the jobs and the scheduling information that is calculated by the processing requirements of jobs and the calculating speed of cloud nodes. It is usually easy to obtain information about the calculating speed of the cloud nodes but quite challenging to determine the processing requirements of jobs. To address this problem, the processing requirements of jobs are dynamically estimate according to the job lengths from user application specifications or historical data [14,8]. The job scheduler maps the job to specific time intervals on the cloud nodes according to the state s of jobs, the states of cloud nodes, the scheduling information of jobs. The job scheduling is to map job to cloud node according to the object of the optimization. There are several criteria that are used to evaluate the efficiency and the effective of job scheduling algorithm. The most popular of which are makespan and flowtime.

In order to formalize the problem, we explain the definition of the terminology relevant to our objective problem. A cloud node (CN) is a set of computational resources with limited computation speed on the cloud. A job is considered as a single set of user applications. A schedule is the mapping of the jobs to specific time intervals on the CN. Assumption the job set is $J = \{J_1, J_2, \ldots J_n\}$, for any $i, j (i \neq j), J_i$ and J_j are irrelevant and preemption is not allowed. Each job J_j has its corresponding length (processing requirement). The set of job length is $L = \{l_1, l_2, \ldots, l_n\}$, l_j is the length of job J_j. The unit of l_j $(j = 1, 2, \ldots, n)$ is the number of cycles. The CN set is $C = \{C_1, C_2, \ldots, C_m\}$. Each CN has corresponding calculating speed. The set of CN speed is $S = \{s_1, s_2, \ldots, s_m\}$. The unit of s_i $(i = 1, 2, \ldots, m)$ is the number of Cycles Per Unit Time (CPUT). s_i is the speed of CN C_i. Individual jobs J_j $(j = 1, 2, \ldots, n)$ must be processed until completion on a single CN. We define $A = (a_{ij})$ $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ as the time it takes CN C_i to complete job J_j.

$$a_{ij} = \frac{l_j}{s_i} \tag{1}$$

The binary assignment matrix $X = (x_{ij})$ $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ is used to denote decision variables.

$$x_{ij} = \begin{cases} 1 & \text{if } J_j \text{ is assigned to cloud node } C_i \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

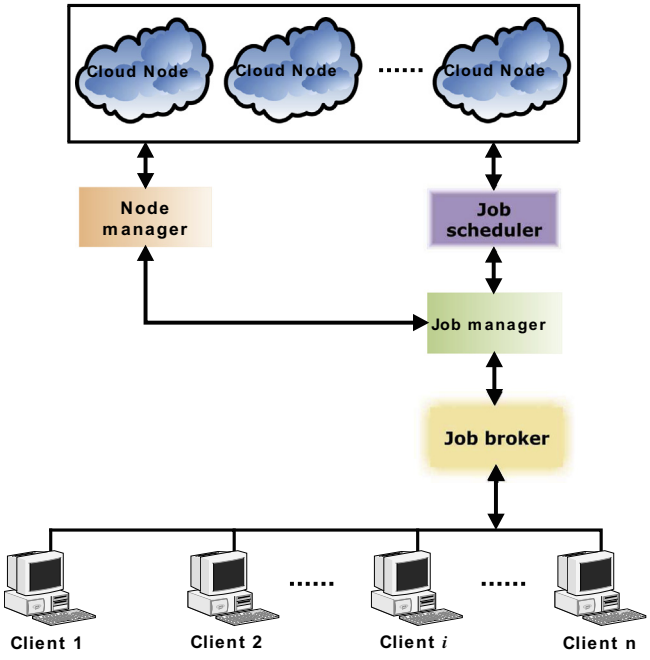

Fig. 1. General framework of cloud and job systems.

Because each job can only be executed on one cloud node, the following equation must be satisfied.

$$\sum_{j=1}^{n} x_{i,j} = 1 \qquad \forall i \tag{3}$$

In order to simplify the model, the setup and transfer times are ignored. The job scheduling is to map J_j $(j = 1, 2, \ldots, n)$ to C_i $(i = 1, 2, \ldots, m)$ so as to minimize the completion time. For example, we schedule 13 jobs $(J_1 \sim J_{13})$ on 3 nodes (C_1, C_2, C_3). The set of CN speed is $S = \{4, 3, 2\}$ and the set of job length is $L = \{6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60\}$. In Fig. 2(a), the jobs J_3,J_4,J_6,J_7,J_8 and J_9 are allocated on cloud node 1; J_5,J_{11} and J_{13} are allocated on cloud node 2; J_1,J_2,J_{10} and J_{12} are allocated on cloud node 3. The job execution time $a_{1,3} = 4, a_{1,4} = 5, a_{1,6} = 7, a_{1,7} = 7.5, a_{1,8} = 9, a_{1,9} = 10$; $a_{2,5} = 8, a_{2,11} = 16$, $a_{2,13} = 20$; $a_{3,1} = 3, a_{3,2} = 6, a_{3,10} = 21, a_{3,12} = 26$. For cloud node 1, its single node flowtime $\sum_{j=1}^{N} a_{1,j} = 42.5$. The other two cloud node flowtimes are 44 and 56, respectively. In job scheduling, the longest single node flowtime is called as makespan, and the sum of all single node flowtimes is the schedule flowtime. So the makespan is 56 and the schedule flowtime is 142.5. An optimal schedule is illustrated in Fig. 2(b), in which the jobs J_3,J_7,J_8,J_{10} and J_{13} are allocated on cloud node 1; J_1,J_4,J_5,J_9 and J_{11} are allocated on cloud node 2; J_2,J_6 and J_{12} are allocated on cloud node 3. The job execution time $a_{1,3} = 4, a_{1,7} = 7.5, a_{1,8} = 9, a_{1,10} = 10.5, a_{1,13} = 15$; $a_{2,1} = 2, a_{2,4} = 6.6667, a_{2,5} = 8, a_{2,9} = 13.3333, a_{2,11} = 16$; $a_{3,2} = 6, a_{3,6} = 14, a_{3,12} = 26$. The single node flowtime of cloud node 1 is $\sum_{j=1}^{N} a_{1,j} = 46$. The other two cloud node flowtimes are also 46. In this optimal schedule solution, the makespan is 46 and the schedule flowtime is 138. Makespan *MS* is the longest one among all single node flowtimes, and the schedule flowtime *SF* is the sum of the finalization time of all jobs. In job shop scheduling, a weighted aggregation is often used to achieve a balance between these two metrics, that is:

$$f = w_1 * MS + w_2 * SF \tag{4}$$

where w_1 and w_2 are non-negative weights and $w_1 + w_2 = 1$. The weights can either be fixed or adapted dynamically during the optimization process. Unfortunately, $MS \ll SF$ in a large-scale cloud environment. In other words, *MS* would vanish out of *SF* unless w_1 is much greater than w_2. So $w_1 = 1$ and $w_2 = 0$ for job scheduling in cloud computing. An optimal schedule will be one that optimizes the makespan, as illustrated in Fig. 2(b). Our job scheduling objective is to minimize the makespan in this article.

For any schedule *X*, the makespan *MS* is formalized as follows.

$$MS(X) = \max_{i \in 1,2,\ldots,m} \sum_{j=1}^{n} a_{i,j} \times x_{i,j} \tag{5}$$

The objective of the job scheduling is to search the schedule *X* that minimizes the *MS* while satisfying the schedule feasibility constraints. That is to say, the problem is formalized as follows.

$$Obj: \ f(X) = \min MS(X) = \min \max_{i \in \{1,2,\cdots,m\}} \sum_{j=1}^{n} a_{i,j} \times x_{i,j} \tag{6}$$

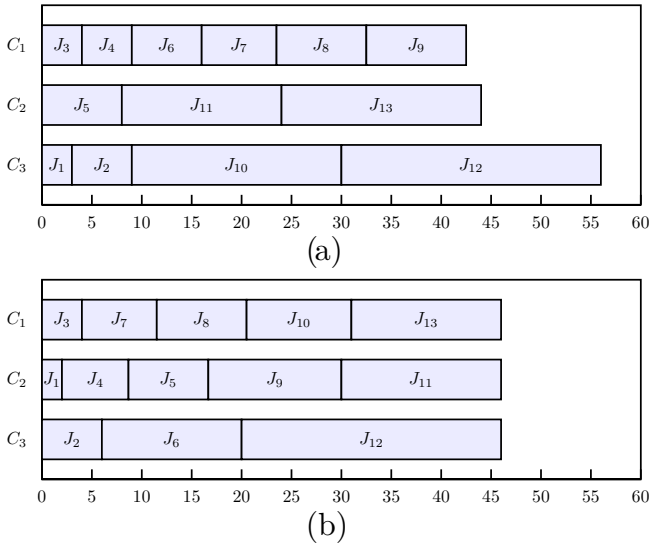

Fig. 2. An example about makespan and flowtime.

s.t.

$$x_{i,j} \in \{0,1\} \ (i = 1,2,\ldots,m;\ j = 1,2,\ldots,n) \tag{7}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad \forall i \tag{8}$$

4. Biogeography-based optimization

In this section, we present the biogeography-based optimization (BBO) for the job scheduling problems. Firstly, the basic BBO is summarized in Algorithm 1. And then our BBO1 is presented as a baseline algorithm in Algorithm 2 to solve the discrete job scheduling problem in cloud computing. We further upgrade the baseline algorithm BBO1 to BBO2 in Algorithm 3.

4.1. Biogeography-based optimization

BBO is a population-based optimization algorithm and maintains its set of solutions from one iteration to the next, relying on migration to probabilistically adapt those solutions. On the other hand, genetic algorithm and evolutionary strategies are reproductive strategies and ant colony optimization generates a new set of solutions with each iteration. In BBO approach, solutions are maintained from one iteration to the next, but each solution is able to learn from its neighbors and adapt itself as the algorithm progresses [16]. Algorithm 1 is the one generation of standard BBO [17].

Algorithm 1. BBO Algorithm

1: Initialize the BBO parameters.
There are some parameters to set which are n (habitat size), I (maximum immigration rate), E (maximum emigration rate), e (number of elites to retain for elitism), and m_{max} (maximum mutation probability).

2: Initialize a random set of habitats, each habitat corresponding to a potential solution the given problem. The population of habitats is initialized randomly using binary habitat representation. These habitats are not optimal but a valid. Each job for the feasible habitats is assigned to only one cloud node.

3: For each habitat, map the HSI (habitat suitability index) to the number of species, the immigration rate using Eq. (9), and the emigration rate using Eq. (10), where s_k is the number of species for habitat k, and s_{max} is the maximum number of species.

$$\lambda_k = I\left(1 - \frac{s_k}{s_{max}}\right) \tag{9}$$

$$\mu_k = \frac{E s_k}{s_{max}} \tag{10}$$

4: Probabilistically use immigration and emigration to modify each non-elite habitat.

5: For each habitat, update the probability of its species count. Then, mutate each non-elite habitat based on its probability using Eq. (11), where P_s is the probability that a habitat contains exactly s species, and P_{max} is the probability that a habitat contains maximum of species.

$$m(s) = m_{max}\left(1 - \frac{P_s}{P_{max}}\right) \tag{11}$$

6: Go to step 3 for the next iteration. This loop can be terminated after a predefined number of generations or after an acceptable problem solution has been found.

4.2. Improved BBO for Job Scheduling in Cloud Computing

We present the BBO1 in Algorithm 2 for the discrete job scheduling problem in cloud computing by introducing the standard BBO of Algorithm 1. Each habitat should have only one Cloud node for each job for its feasibility of solution considering Eq. (3) when initializing the population of solutions which are the two dimensional discrete representation with discrete decision variables. This feasibility should be followed when generating the neighbors of habitats. BBO1 generates new habitats by migration using randomly selected habitats. BBO1 updates the habitats anytime.

Our BBO1 is improved further into BBO2 in Algorithm 3, which generates new habitats by migration using the best habitats for convergence. The BBO2 also updates the habitats if new generated habitat is better than old one after migration for converged search and after mutation for diversified search to balance the harmony between exploitation and exploration.

Algorithm 2. BBO1 Algorithm for Job Scheduling in Cloud Computing

1: Initialize the parameters (n, I, E, e, m_{max}).
2: Initialize a population of habitats (non-optimal but a valid) randomly using binary habitat representation. The feasible habitats have only one Cloud for each job.
3: $cycle \leftarrow 1$
4: **repeat**
5: Produce the new habitats by applying the immigration rate λ_k and emigration rate μ_k for each non-elite habitat. The column of each job should be exchanged in migration (immigration and emigration) for habitat's feasibility.
6: Evaluate the population using Eqs. (6)–(8).
7: Update the habitats.
8: Produce the new habitats by applying the mutation for each habitat. Select any job column based on mutation rate $m(s)$ and change the assigned Cloud to job randomly. The feasible habitats have only one Cloud for each job.
9: Evaluate the population using Eqs. (6)–(8).
10: Update the habitats.
11: Memorize the best habitat achieved so far.
12: $cycle \leftarrow cycle + 1$
13: **until** $cycle = MCN$

Algorithm 3. BBO2 Algorithm for Job Scheduling in Cloud Computing

1: Initialize the parameters (n, I, E, m_{max}).
2: Initialize a population of habitats (non-optimal but a valid) randomly using binary habitat representation. The feasible habitats have only one Cloud for each job.
3: $cycle \leftarrow 1$
4: **repeat**
5: Produce the new habitats by applying the immigration rate λ_k and emigration rate μ_k for each non-elite habitat. These new habitats are immigrated from the best habitats. The column of each job should be exchanged in migration (immigration and emigration) for habitat's feasibility.
6: Evaluate the population using Eqs. (6)–(8).
7: Update the habitats if new generated habitat is better than old one.
8: Produce the new habitats by applying the mutation for each habitat. Select any job column based on mutation rate $m(s)$ and change the assigned Cloud node to job randomly. The feasible habitats have only one cloud node for each job.
9: Evaluate the population using Eqs. (6)–(8).
10: Update the habitats if new generated habitat is better than old one.
11: Memorize the best habitat achieved so far.
12: $cycle \leftarrow cycle + 1$
13: **until** $cycle = MCN$

5. Experiments and analysis

5.1. Experimental settings

The experiment simulation was run on Intel® Core™2 Duo CPU (2.66 GHz, 2G RAM) in this paper. We have used 7 different job scheduling problems $P_{11}(3,13)$, $P_{21}(5,100)$, $P_{31}(8,60)$, $P_{41}(10,50)$, $P_{51}(10,100)$, $P_{61}(60,500)$, and $P_{71}(100,1000)$ which Liu et al. [34] made. We used three more instances ($P_{12} \sim P_{72}, P_{13} \sim P_{73}, P_{14} \sim P_{74}$) for every size which Kim et al. [8] made. Each cloud node has a non-uniform capacity, which is constant along the period of simulation. Each job has also a constant job length. Job scheduling is dependent on the capacity of could node and the length of the jobs assigned to it. The completion time of each job would be varying, when the job is assigned to different nodes in different scheduling solutions. Our objective is to complete all the jobs to minimize the makespan. The specific parameter setting of BBO is described in Table 1 using several experiments.

5.2. Results and discussion

For small scale job scheduling problem, the speeds of 3 cloud nodes are 4, 3, 2 CPUT. The job lengths of 13 jobs are 6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60 cycles. Each completion time of each job can be different because there is different capacity for each cloud node and each job has different job length. Fig. 3 is the trend of convergence for job scheduling (3,13)

Table 1
Parameter setting of BBO.

Parameter name	Value
Habitat size n	50
Maximum immigration rate I	0.5
Maximum emigration rate E	1
Number of elites to retain for elitism e	1
Maximum mutation probability m_{max}	0.1

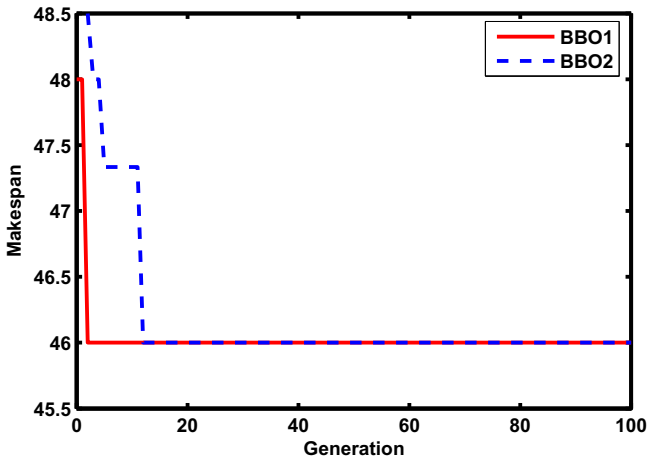

Fig. 3. Convergence trend for (3,13) using BBO1 and BBO2.

using BBO1 and BBO2. The optimal makespan is 46 that the total job length of each cloud node is divided by the speed of each cloud.

Further, we considered the algorithms for other six (G,J) pairs, i.e. (5,100), (8,60), (10,50), (10,100), (60,500), and (100,1000). Figs. 4–9 is the trend of convergence for job scheduling problems using BBO1 and BBO2.

The best makespans of job scheduling are 85.536, 41.668, 35.292, and 59.269 for (5,100), (8,60), (10,50), and (10,100) using BBO1 and 50.942 and 63.114 for (60,500) and (100,1000) using BBO2, respectively. The best solutions are shown in Tables 2–5 for the (5,100), (8,60), (10,50), and (10,100).

The performance of BBO1 and BBO2 are compared in Tables 6–9. The performance of these two methods are almost the same for the small and middle size of problems. The performance of BBO2 is much better than that of BBO1 for the large size job scheduling (60,500) and (100,1000) problems.

Table 10 is the comparison of simulation results using BBO with early study of Liu et al. [34] and Kim et al. [8]. The average value and standard deviation of makespan using BBO are more competitive comparing to the genetic algorithm (GA),

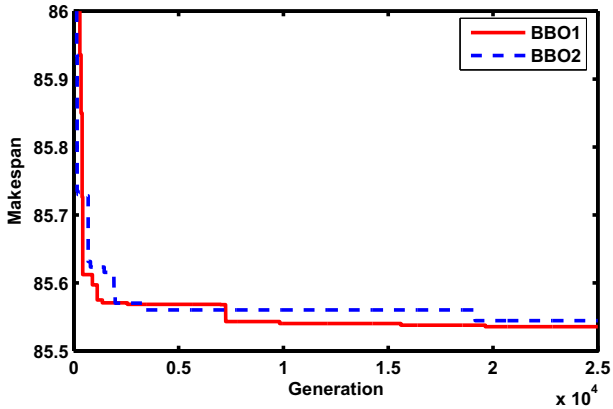

Fig. 4. Convergence trend for (5,100) using BBO1 and BBO2.

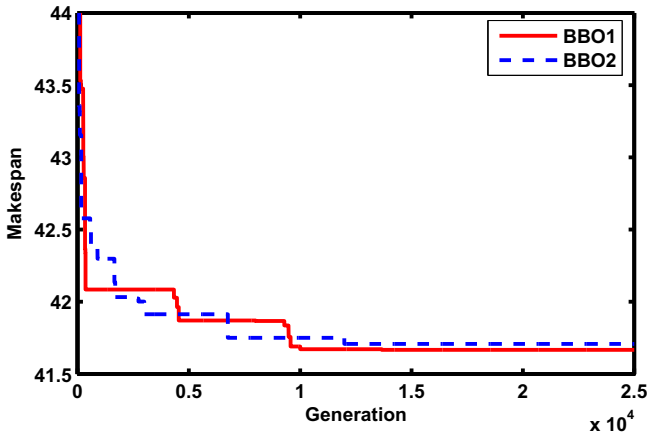

Fig. 5. Convergence trend for (8,60) using BBO1 and BBO2.

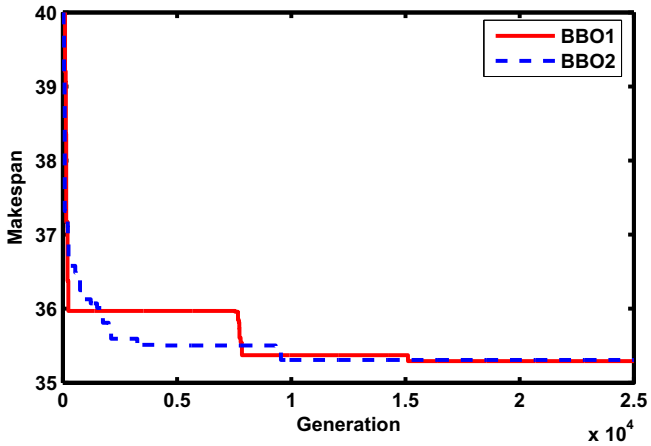

Fig. 6. Convergence trend for (10,50) using BBO1 and BBO2.

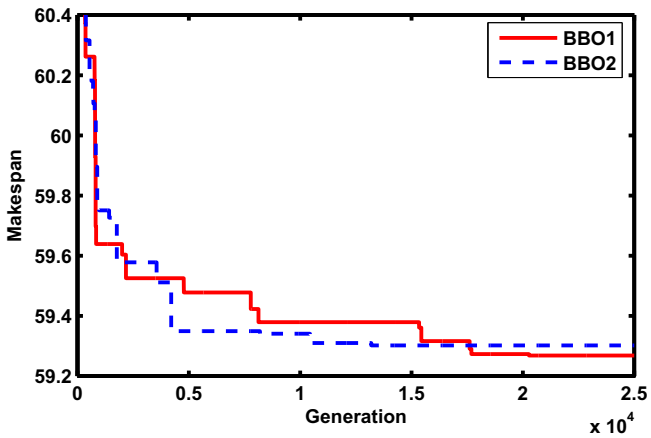

Fig. 7. Convergence trend for (10,100) using BBO1 and BBO2.

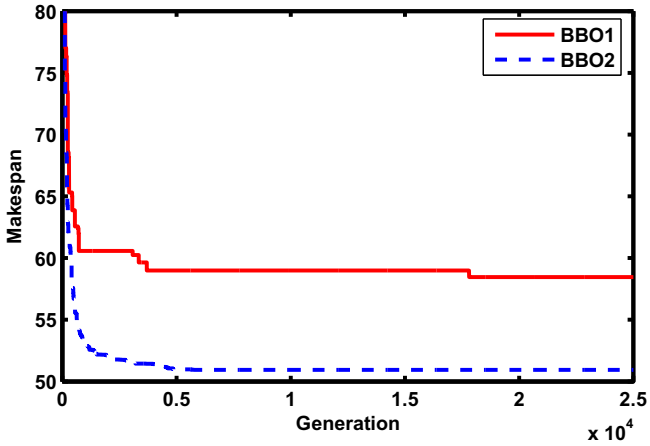

Fig. 8. Convergence trend for (60,500) using BBO1 and BBO2.

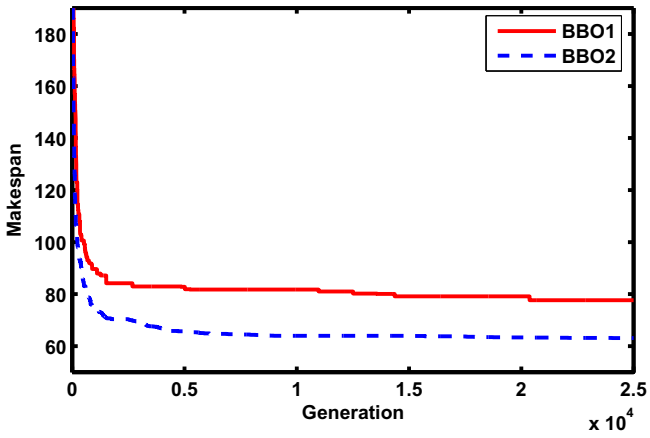

Fig. 9. Convergence trend for (100,1000) using BBO1 and BBO2.

Table 2
Best solutions for job scheduling problem (5,100).

Cloud node	Assigned job number dataset
1	7, 9, 10, 13, 14, 19, 23, 36, 41, 45, 47, 54, 56, 61, 62, 70, 83, 84, 97, 98
2	20, 24, 63, 69, 82, 86, 88, 89, 94, 95, 99, 100
3	8, 15, 16, 22, 33, 40, 43, 48, 52, 55, 77, 79, 80, 81, 85, 87, 90,
4	1, 2, 3, 4, 11, 18, 29, 30, 32, 39, 42, 46, 49, 57, 58, 59, 64, 65, 71, 72, 75, 76, 91, 93, 96
5	5, 6, 12, 17, 21, 25, 26, 27, 28, 31, 34, 35, 37, 38, 44, 50, 51, 53, 60, 66, 67, 68, 73, 74, 78, 92

Table 3
Best solutions for job scheduling problem (8,60).

Cloud node	Assigned job number dataset
1	5, 6, 14, 17, 19, 28, 29, 42, 43, 53, 59
2	7, 16, 31, 52, 55
3	1, 11, 15, 25, 30, 35, 38, 41, 46, 54
4	4, 23, 24, 32, 33, 34
5	10, 12, 13, 26, 27, 47, 48, 56, 58, 60
6	3, 9, 21, 22, 37, 40, 44, 45, 49, 50
7	2, 8, 20, 39, 51
8	18, 36, 57

Table 4
Best solutions for job scheduling problem (10,50).

Cloud node	Assigned job number dataset
1	1, 11, 18, 26, 42
2	4, 23, 24, 33
3	5, 15, 22, 32, 44, 48, 49
4	3, 19, 28
5	7, 16, 35, 40
6	8, 21, 36, 45
7	2, 31, 34, 39, 46, 50
8	6, 12, 14, 27, 29, 37
9	9, 10, 13, 25, 30, 43
10	17, 20, 38, 41, 47

Table 5
Best solutions for job scheduling problem (10,100).

Cloud node	Assigned job number dataset
1	2, 25, 30, 32, 35, 36, 43, 47, 65, 77, 82, 87, 89, 90, 93
2	4, 5, 10, 18, 24, 31, 59, 64, 67
3	6, 29, 39, 40, 62, 70, 85, 86, 92
4	1, 7, 9, 16, 19, 20, 41, 42, 48, 55, 56, 61, 66, 76, 78, 88
5	13, 22, 27, 28, 44, 46, 60, 72, 81, 83, 84, 91
6	12, 14, 26, 33, 49, 53, 73, 75, 100
7	11, 17, 23, 34, 37
8	21, 38, 45, 54, 58, 69, 71, 74, 79, 94, 98
9	52, 57, 63, 68, 95, 97, 99
10	3, 8, 15, 50, 51, 80, 96

Table 6
Performance comparison of BBO1 and BBO2 using Liu's dataset [14] ($P_{11} \sim P_{71}$).

P[a]	Algorithms	min[b]	$\overline{S}$[c]	σ[d]
P_{11}	BBO1	46.000	46.350	0.337
	BBO2	46.000	46.250	0.354
P_{21}	BBO1	85.536	85.551	0.014
	BBO2	85.545	85.561	0.015
P_{31}	BBO1	41.668	41.770	0.097
	BBO2	41.709	41.784	0.055
P_{41}	BBO1	35.292	35.459	0.145
	BBO2	35.307	35.534	0.259
P_{51}	BBO1	59.269	59.330	0.084
	BBO2	59.302	59.343	0.036
P_{61}	BBO1	58.450	59.030	0.348
	BBO2	50.942	51.230	0.287
P_{71}	BBO1	77.614	79.738	1.022
	BBO2	63.114	63.335	0.241

[a] Scheduling problem.
[b] Minimum.
[c] Average.
[d] Standard deviation.

simulated annealing (SA), and particle swarm optimization (PSO) for the job scheduling problems with early study of Liu et al. [34].

Table 11 is the comparative study of BBO with ABC for the job scheduling problems with early study of Kim et al. [8]. The average value and standard deviation of makespan using biogeography-based optimization (BBO) are more competitive comparing to the artificial bee colony (ABC) with less standard deviation for large size job scheduling (60,500) and (100,1000) problems.

In order to assess the statistical significance of the performance differences between BBO1 and BBO2 as shown in Table 12 or between ABC and BBO as shown in Table 13 the results were evaluated using Mann–Whitney [35] tests by IBM SPSS

Table 7
Performance comparison of BBO1 and BBO2 using ($P_{12} \sim P_{72}$).

P[a]	Algorithms	min[b]	$\bar{S}$[c]	σ[d]
P_{12}	BBO1	84.750	85.292	0.322
	BBO2	84.750	85.033	0.158
P_{22}	BBO1	96.919	96.947	0.023
	BBO2	96.933	96.968	0.022
P_{32}	BBO1	35.867	35.908	0.040
	BBO2	35.870	35.915	0.030
P_{42}	BBO1	39.879	40.020	0.107
	BBO2	39.943	40.112	0.181
P_{52}	BBO1	68.727	68.782	0.039
	BBO2	68.695	68.848	0.070
P_{62}	BBO1	53.304	53.847	0.362
	BBO2	46.395	46.595	0.135
P_{72}	BBO1	78.232	79.775	0.959
	BBO2	63.149	63.357	0.164

[a] Scheduling problem.
[b] Minimum.
[c] Average.
[d] Standard deviation.

Table 8
Performance comparison of BBO1 and BBO2 using ($P_{13} \sim P_{73}$).

P[a]	Algorithms	min[b]	$\bar{S}$[c]	σ[d]
P_{13}	BBO1	77.250	77.742	0.313
	BBO2	77.250	77.400	0.156
P_{23}	BBO1	99.360	99.372	0.013
	BBO2	99.393	99.403	0.009
P_{33}	BBO1	35.424	35.485	0.047
	BBO2	35.467	35.518	0.041
P_{43}	BBO1	38.287	38.486	0.164
	BBO2	38.399	38.524	0.151
P_{53}	BBO1	63.233	63.352	0.139
	BBO2	63.329	63.390	0.053
P_{63}	BBO1	54.107	55.221	0.537
	BBO2	47.621	47.973	0.267
P_{73}	BBO1	81.147	82.422	1.041
	BBO2	65.315	65.682	0.282

[a] Scheduling problem.
[b] Minimum.
[c] Average.
[d] Standard deviation.

software to make a decision to accept or reject a hypothesis. The Mann–Whitney test is usually used to evaluate two different data populations, such as performance results from two separate algorithms [36,37].

- Mann–Whitney Test for Table 12:
  - H_0 : There is no performance difference between BBO1 and BBO2.
  - H_1 : There is a performance difference between BBO1 and BBO2.

- Mann–Whitney Test for Table 13:
  - H_0 : There is no performance difference between ABC and BBO.
  - H_1 : There is a performance difference between ABC and BBO.

The results of the non-parametric tests for a 95% significance level ($p < 0.05$) are shown in Table 12. These indicate that there are no statistically significant performance difference between BBO1 and BBO2 for the $P_{11}, P_{21}, P_{31}, P_{41}$, and P_{51}. There are statistically significant differences in performance between BBO1 and BBO2 for problems P_{61} and P_{71} as shown in

Table 12. BBO2 is much better than BBO1 for P_{61} and P_{71}. We have almost the same results for the $P_{12} \sim P_{72}, P_{13} \sim P_{73}$, and $P_{14} \sim P_{74}$. The overall trend of differences become significant as job scheduling problems increase in sizes. We also do the statistical tests to compare the performance difference of our proposed BBO with artificial bee colony (ABC) proposed by Kim et al. [8] as shown in Table 13. There are statistically significant performance difference between ABC and BBO. ABC is slightly better than BBO for P_{11}, P_{21}, P_{31}, P_{41}, and P_{51}. But, BBO is much better than ABC for P_{61} and P_{71}.

In the BBO paradigm we find a habitat (solution) which represents a position in the problem space. Since the positions are in general different, the habitat group has the capability to explore the space searching for better solutions. The term *Diversity* is introduced to quantify the diversity of the habitat group and is defined as follows:

$$Diversity = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{1}{d}\sum_{j=1}^{d}(x_{ij} - \bar{x}_j)^2} \tag{12}$$

Here, n is the habitat group size, d is the dimension of the search space (coordinates of the food sources), $\bar{x}$ is the centre point of the habitat, and its jth dimension is denoted by $\bar{x}_j$. Fig. 10 shows the typical evolution of habitat diversity for each of the two algorithms during an optimization run. In the BBO2 algorithm the diversity undergoes a fluctuating decrease phase before stabilizing at a very low value after 500 generations. Two different phases can be observed with the BBO1 algorithm. The diversity decreases, but stabilizes after 200 generations at a low state of diversity. Considering Fig. 10, it can be seen that our algorithms initially provide large habitat group diversity facilitating exploration of the global solution space and then focuses on the current best solution with a refining step to achieve local exploitation. Experimental results illustrate that the algorithm achieves the best balance between global exploration and local exploitation.

Table 9
Performance comparison of BBO1 and BBO2 using ($P_{14} \sim P_{74}$).

P[a]	Algorithms	min[b]	$\bar{S}$[c]	σ[d]
P_{14}	BBO1	66.250	66.625	0.315
	BBO2	66.250	66.383	0.105
P_{24}	BBO1	99.545	99.562	0.019
	BBO2	99.560	99.571	0.011
P_{34}	BBO1	38.632	38.711	0.048
	BBO2	38.686	38.728	0.029
P_{44}	BBO1	39.290	39.501	0.206
	BBO2	39.393	39.493	0.065
P_{54}	BBO1	70.245	70.322	0.074
	BBO2	70.365	70.411	0.038
P_{64}	BBO1	58.223	58.529	0.394
	BBO2	50.522	50.822	0.369
P_{74}	BBO1	80.104	81.157	0.604
	BBO2	64.199	64.620	0.229

[a] Scheduling problem.
[b] Minimum.
[c] Average.
[d] Standard deviation.

Table 10
Comparative study of BBO with GA, SA and PSO of Liu et al. [14] and ABC of Kim et al. [8].

P[a]	GA		SA		PSO		ABC best		BBO best	
	$\bar{S}$[b]	σ[c]	$\bar{S}$[b]	σ[c]	$\bar{S}$[b]	σ[c]	$\bar{S}$[b]	σ[c]	$\bar{S}$[b]	σ[c]
P_{11}	47.117	0.770	46.600	0.486	46.267	0.286	46.000	0.000	46.250	0.354
P_{21}	85.743	0.622	90.734	6.383	84.054	0.503	85.530	0.001	85.551	0.014
P_{31}	42.927	0.415	55.459	2.061	41.949	0.694	41.598	0.003	41.770	0.097
P_{41}	38.043	0.661	41.789	8.077	37.667	0.607	35.190	0.009	35.459	0.145
P_{51}	63.149	0.373	70.549	7.414	62.033	0.881	59.182	0.004	59.330	0.084
P_{61}	55.587	0.607	65.489	7.046	54.794	0.852	52.946	0.665	51.230	0.287
P_{71}	73.105	0.369	83.762	7.103	72.970	0.606	70.660	4.482	63.335	0.241

[a] Scheduling problem.
[b] Average.
[c] Standard deviation.

Table 11
Comparative study of BBO with ABC of Kim et al. [8] for the job scheduling problems.

P[a]	Algorithms							
	ABC				BBO			
	ABC best	min[b]	$\bar{S}$[c]	σ[d]	BBO best	min[b]	$\bar{S}$[c]	σ[d]
P_{11}	BABC, EBABC1, EBABC2	46.000	46.000	0.000	BBO2	46.000	46.250	0.354
P_{12}	BABC, EBABC2	84.750	84.775	0.079	BBO2	84.750	85.033	0.158
P_{13}	BABC	77.250	77.250	0.000	BBO2	77.250	77.400	0.156
P_{14}	BABC, EBABC2	66.250	66.250	0.000	BBO2	66.250	66.383	0.105
P_{21}	EBABC2	85.529	85.530	0.001	BBO1	85.536	85.551	0.014
P_{22}	EBABC2	96.914	96.915	0.001	BBO1	96.919	96.947	0.023
P_{23}	EBABC2	99.355	99.356	0.001	BBO1	99.360	99.372	0.013
P_{24}	EBABC2	99.536	99.537	0.001	BBO1	99.545	99.562	0.018
P_{31}	EBABC2	41.592	41.598	0.003	BBO1	41.668	41.770	0.097
P_{32}	EBABC2	35.817	35.822	0.005	BBO1	35.867	35.908	0.040
P_{33}	EBABC2	35.383	35.388	0.003	BBO1	35.424	35.485	0.047
P_{34}	EBABC1	38.571	38.577	0.004	BBO1	38.632	38.711	0.048
P_{41}	EBABC2	35.174	35.190	0.009	BBO1	35.292	35.459	0.145
P_{42}	EBABC2	39.674	39.687	0.014	BBO1	39.879	40.020	0.107
P_{43}	EBABC1	38.124	38.139	0.013	BBO1	38.287	38.486	0.164
P_{44}	EBABC2	38.983	39.007	0.015	BBO2	39.393	39.493	0.065
P_{51}	EBABC2	59.175	59.182	0.004	BBO1	59.269	59.330	0.084
P_{52}	EBABC2	68.634	68.638	0.004	BBO1	68.727	68.782	0.039
P_{53}	EBABC1	63.165	63.173	0.006	BBO1	63.233	63.351	0.139
P_{54}	EBABC1	70.096	70.106	0.006	BBO1	70.245	70.322	0.074
P_{61}	EBABC2	52.131	52.946	0.665	BBO2	50.942	51.230	0.287
P_{62}	EBABC2	47.285	48.451	1.158	BBO2	46.395	46.595	0.135
P_{63}	EBABC2	48.492	49.688	0.895	BBO2	47.621	47.973	0.267
P_{64}	EBABC2	51.414	53.121	0.895	BBO2	50.522	50.822	0.369
P_{71}	EBABC1	65.109	70.660	4.482	BBO2	63.114	63.335	0.241
P_{72}	EBABC1	68.227	70.503	1.907	BBO2	63.149	63.357	0.164
P_{73}	EBABC1	69.561	74.360	3.811	BBO2	65.315	65.682	0.282
P_{74}	EBABC1	69.054	72.430	3.065	BBO2	64.199	64.619	0.229

[a] Scheduling problem.
[b] Minimum.
[c] Average.
[d] Standard deviation.

Table 12
Difference of BBO1 and BBO2 using Mann–Whitney test.

P[a]	Group	Result
P_{11}	BBO1, BBO2	Accept H_0
P_{21}	BBO1, BBO2	Accept H_0
P_{31}	BBO1, BBO2	Accept H_0
P_{41}	BBO1, BBO2	Accept H_0
P_{51}	BBO1, BBO2	Accept H_0
P_{61}	BBO1, BBO2	Accept H_1, BBO1 > BBO2
P_{71}	BBO1, BBO2	Accept H_1, BBO1 > BBO2

[a] Scheduling problem.

Table 13
Performance difference of ABC, BBO using Mann–Whitney test.

P[a]	Group	Result
P_{11}	EBABC2, BBO2	Accept H_1, EBABC2 < BBO2
P_{21}	EBABC2, BBO1	Accept H_1, EBABC2 < BBO1
P_{31}	EBABC2, BBO1	Accept H_1, EBABC2 < BBO1
P_{41}	EBABC2, BBO1	Accept H_1, EBABC2 < BBO1
P_{51}	EBABC2, BBO1	Accept H_1, EBABC2 < BBO1
P_{61}	EBABC2, BBO2	Accept H_1, EBABC2 > BBO2
P_{71}	EBABC1, BBO2	Accept H_1, EBABC1 > BBO2

[a] Scheduling problem.

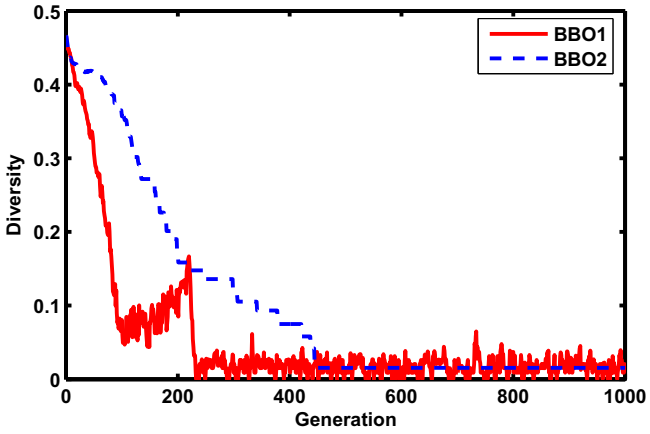

Fig. 10. Comparison of the evolution of habitat diversity.

6. Conclusions

The biogeography-based optimization (BBO) is relatively new and has the advantage of adapting new good solutions. This BBO adaptive process is compared to the genetic algorithm and other heuristic algorithms which are reproductive process. The BBO1 is developed using standard BBO to find the optimal job schedule of cloud computing in this paper. We also develop BBO2. The BBO2 produces the new habitats by applying the immigration and emigration for each non-elite habitat from the several best habitats and updates the habitats if new generated habitat is better than old one. The simulation results of BBO1 and BBO2 is compared for 4 different data sets for 7 different sizes. The BBO1 is slightly better than BBO2 for the small and middle sizes of problems. But, the BBO2 is much better than BBO1 for the large sizes of problems. The performance of BBO is much better than those of genetic algorithm, simulated annealing, and particle swarm optimization when the problems are large. ABC is slightly better than BBO in the middle sizes of problems. But, BBO is better than ABC in the large sizes of problems.

Acknowledgments

The authors would like to thank Professor Ajith Abraham for his comments and discussions on the earlier versions of this paper. This study is partly supported by Kangwon National University, the National Natural Science Foundation of China (Grant Nos. 61173035, 61472058) and the Program for New Century Excellent Talents in University (Grant No. NCET-11-0861).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.amc.2014.09.008.

References

[1] Y. Wei, M. Blake, Service-oriented computing and cloud computing: challenges and opportunities, IEEE Internet Comput. 14 (6) (2010) 72–75.
[2] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: Proceedings of the 2008 Grid Computing Environments Workshop, IEEE, 2008, pp. 1–10.
[3] B. Xu, C. Zhao, E. Hu, B. Hu, Job scheduling algorithm based on Berger model in cloud environment, Adv. Eng. Software 42 (7) (2011) 419–425.
[4] S. Ostermann, R. Prodan, T. Fahringer, Extending grids with cloud resource management for scientific computing, in: Proceedings of 10th IEEE/ACM International Conference on Grid Computing, IEEE, 2009, pp. 42–49.
[5] L.F. Bittencourt, E.R. Madeira, N.L. Da Fonseca, Scheduling in hybrid clouds, IEEE Commun. Mag. 50 (9) (2012) 42–47.
[6] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., 1979.
[7] J. Wu, X. Xu, P. Zhang, C. Liu, A novel multi-agent reinforcement learning approach for job scheduling in grid computing, Future Gener. Comput. Syst. 27 (5) (2011) 430–439.
[8] S. Kim, J. Byeon, H. Liu, A. Abraham, S. Mcloone, Optimal job scheduling of grid computing using efficient binary artificial bee colony, Soft Comput. 17 (5) (2013) 867–882.
[9] M. Clerc, Particle Swarm Optimization, Wiley-ISTE, 2006.
[10] A. Yadav, K. Deep, Shrinking hypersphere based trajectory of particles in PSO, Appl. Math. Comput. 220 (2013) 246–267.
[11] H. Liu, A. Abraham, M. Clerc, Chaotic dynamic characteristics in swarm intelligence, Appl. Soft Comput. 7 (3) (2007) 1019–1026.
[12] S. Krause, R. James, J.J. Faria, G.D. Ruxton, J. Krause, Swarm intelligence in humans: diversity can trump ability, Animal Behaviour 81 (5) (2011) 941–948.
[13] E. Cuevas, F. Sención-Echauri, D. Zaldivar, M. Pérez-Cisneros, Multi-circle detection on images using artificial bee colony (ABC) optimization, Soft Comput. 16 (2) (2012) 1–16.

[14] H. Liu, A. Abraham, A. Hassanien, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, Future Gener. Comput. Syst. 26 (8) (2010) 1336–1343.
[15] L. Babu, P.V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, Appl. Soft Comput. 13 (5) (2013) 2292–2303.
[16] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[17] D. Simon, R. Rarick, M. Ergezer, D. Du, Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms, Inf. Sci. 181 (7) (2011) 1224–1248.
[18] P. Brucker, Scheduling Algorithms, Springer Verlag, 2007.
[19] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer Verlag, 2012.
[20] D.-M. Lei, Minimizing makespan for scheduling stochastic job shop with random breakdown, Appl. Math. Comput. 218 (24) (2012) 11851–11858.
[21] S. Song, K. Hwang, Y. Kwok, et al, Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling, IEEE Trans. Comput. 55 (6) (2006) 703.
[22] A. Tchernykh, U. Schwiegelshohn, R. Yahyapour, N. Kuzjurin, On-line hierarchical job scheduling on grids with admissible allocation, J. Scheduling 13 (5) (2010) 545–552.
[23] S. Chung, F. Chan, H. Chan, A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling, Eng. Appl. Artif. Intell. 22 (7) (2009) 1005–1014.
[24] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671.
[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[26] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Techn. Rep. TR06, Erciyes Univ. Press, Erciyes.
[27] E. Hou, N. Ansari, H. Ren, A genetic algorithm for multiprocessor scheduling, IEEE Trans. Parallel Distrib. Syst. 5 (2) (1994) 113–120.
[28] L. Wang, H.J. Siegel, V.P. Roychowdhury, A.A. Maciejewski, Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach, J. Parallel Distrib. Comput. 47 (1) (1997) 8–22.
[29] D. Simon, M. Ergezer, D. Du, Population distributions in biogeography-based optimization algorithms with elitism, Proceedings of IEEE International Conference on Systems, Man and Cybernetics, IEEE, 2009.
[30] H. Ma, D. Simon, Blended biogeography-based optimization for constrained optimization, Eng. Appl. Artif. Intell. 24 (3) (2011) 517–525.
[31] P. Roy, S. Ghoshal, S. Thakur, Biogeography based optimization for multi-constraint optimal power flow with emission and non-smooth cost function, Expert Syst. Appl. 37 (12) (2010) 8221–8228.
[32] L.M. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 50–55.
[33] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, ACM SIGOPS Operating Syst. Rev. 37 (5) (2003) 164–177.
[34] H. Liu, A. Abraham, V. Snášel, S. McLoone, Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments, Inf. Sci. 192 (2012) 228–243.
[35] H. Mann, D. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1) (1947) 50–60.
[36] R. Lahoz-Beltra, C. Perales-Gravan, A survey of nonparametric tests for the statistical analysis of evolutionary computational experiments, Int. J. Inf. Theor. Appl. 17 (1) (2010) 41–61.
[37] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.