Research Journal of Applied Sciences, Engineering and Technology 8(8): 964-975, 2014
DOI:10.19026/rjaset.8.1058
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: March 29, 2014 Accepted: April 28, 2014 Published: August 25, 2014

Research Article

Scheduling Independent Jobs on Computational Grid using Biogeography Based Optimization Algorithm for Makespan Reduction

[1]S. Selvi and [2]D. Manimegalai
[1]Department of Electronics and Communication Engineering, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur-628215,
[2]Department of Information Technology, National Engineering College, Kovilpatti-628503, Tamilnadu, India

Abstract: Due to the development of information and network technologies, idle computers all over the world can be organized and utilized to enhance the overall computation performance. Grid computing refers to the combination of computer resources from multiple administrative domains used to reach a common goal. Grids offer a way of using the information technology resources optimally inside an organization. As the grid environments facilitate distributed computation, the scheduling of grid jobs has become an important issue. This study introduces a novel approach based on Biogeography Based Optimization algorithm (BBO) for scheduling jobs on computational grid. The proposed approach generates an optimal schedule so as to complete the jobs within a minimum period of time. The performance of the proposed algorithm has been evaluated with Genetic Algorithm (GA), Differential Evolution algorithm (DE), Ant Colony Optimization algorithm (ACO) and Particle Swarm Optimization algorithm (PSO).

Keywords: Biogeography based optimization, grid computing, job scheduling, makespan

INTRODUCTION

Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data and storage or network resources across dynamic and geographically dispersed organization (Foster and Kesselmann, 2004). Grid technologies promise to change the way organizations tackle complex computational problems. Grid computing is an evolving area of computing where standards and technology are still being developed to enable this new paradigm.

Users can share grid resources by submitting computing tasks to grid system. The resources of computational grid are dynamic and it belongs to different administrative domains. The participation of resources may be active or inactive within the grid. Hence it is impossible for anyone to manually assign jobs to computing resources in grids. Therefore grid job scheduling is one of the challenging issues in grid computing. Grid scheduling system selects the resources and allocates the user submitted jobs to appropriate resources in such a way that the user and application requirements are met.

To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. In grid computing, as resources are distributed in multiple domains in the Internet, the computational and storage nodes and the underlying networks connecting them are heterogeneous. Thus the heterogeneity results in different capabilities for job processing and data access. In a Grid, resources are usually autonomous and the grid schedulers do not have full control of the resources. The autonomy results in the diversity in local resource management and access control policies and hence grid scheduler is required to be adaptive to different local policies. Grid schedulers work in a dynamic environment where the performance of available resources is constantly changing. A feasible scheduling algorithm should be able to be adaptive to dynamic behaviors. As grid consists of a large number of heterogeneous computing and storage sites connected via wide area networks, grid scheduler has to select the computation sites of an application according to resource status and performance models. Hence the unique characteristics of grid computing such as heterogeneity and autonomy, performance dynamism and resource selection make the design of scheduling algorithms more challenging.

There are many research efforts aiming at job scheduling on the grid. Scheduling m jobs to n resources with an objective to minimize the total execution time had been shown to be NP-complete (Ibarra and Ki,

Corresponding Author: S. Selvi, Department of Electronics and Communication Engineering, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur-628215, Tamilnadu, India, Tel.: +91 8903484336; Fax: +91 4639 243188

1977). Therefore the use of heuristics is the defacto approach in order to cope in practice with its difficulty. Krauter *et al*. (2002) provided a useful survey on grid resource management systems, in which most of the grid schedulers such as AppLes, Condor, Globus, Legion, Netsolve, Ninf and Nimrod use simple batch scheduling heuristics. Jarvis *et al*. (2003) proposed the scheduling algorithm using metaheuristics and compared First Come First Serve heuristic with genetic algorithm to minimize the makespan and it was found that metaheuristics generate good quality schedules than batch scheduling heuristics. Braun *et al*. (2001) studied the comparison of the performance of batch queuing heuristics, tabu search, genetic algorithm and simulated annealing to minimize the makespan. The results revealed that genetic algorithm achieved the best results compared with batch queuing heuristics. Liu *et al*. (2010) proposed a fuzzy Particle Swarm Optimization algorithm for scheduling jobs on computational grid with the minimization of makespan as the main criterion. They empirically showed that their method outperforms the genetic algorithm and simulated annealing approach. The results revealed that the PSO algorithm has an advantage of high speed of convergence and the ability to obtain faster and feasible schedules. In this study we introduce a novel approach based on Biogeography Based optimization for scheduling jobs on computational grid.

Biogeography Based Optimization (Simon, 2008) is a new evolutionary algorithm for global optimization that was introduced in 2008. BBO is an application of biogeography to evolutionary algorithms. Biogeography is the study of the distribution of biodiversity over space and time. It aims to analyze where organisms live and in what abundance. Biogeography not only gives a description of species distributions, but also a geographical explanation. Biogeography is modeled in terms of such factors as habitat area, immigration rate and emigration rate and describes the evolution, extinction and migration of species. BBO has certain features in common with other population-based optimization methods. Like GA and PSO, BBO can share information between solutions. This makes BBO applicable to many of the same types of problems that GA and PSO are used for, including unimodal, multimodal and deceptive functions. One distinctive feature of BBO is that the original population is not discarded after each generation (Simon, 2008). It is rather modified by migration. Another distinctive feature is that, for each generation, BBO uses the fitness of each solution to determine its immigration and emigration rate. BBO has also demonstrated good performance on various unconstrained bench mark functions (Du *et al*., 2009; Ma *et al*., 2009; Simon, 2008). It has also been applied to real-world optimization problems, including sensor selection (Simon, 2008), economic load dispatch problem (Bhattacharya and Chattopadhyay, 2010), satellite image classification (Panchal *et al*., 2009), rectangular micro strip antenna design (Lohokare *et al*., 2009), design of Yagi-Uda Antenna (Singh *et al*., 2010), traveling salesman problem (Song *et al*., 2010) and robot controller tuning (Lozovyy *et al*., 2011). This is a pioneer effort in the research area of Grid scheduling, which makes use of Biogeography Based optimization technique to dynamically generate an optimum schedule so as to complete the tasks within a minimum period of time as well as utilizing the resources in an efficient way.

METHODOLOGY

The grid job scheduling problem: A computational grid is a hardware and software infrastructure that provides dependable, consistent pervasive and inexpensive access to high end computational capabilities (Foster and Kesselmann, 2004). It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications and data, all connected through the Internet and a middleware layer that provides basic services for security, monitoring, resource management and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to access what and under what conditions (Foster and Iamnitchi, 2003). The real and specific problem that underlies the grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations (Foster *et al*., 2001). For clarity, some key terminologies (Dong and Akl, 2006) are defined as follows.

Grid node: A grid node is an autonomous entity composed of one or multiple resources. The computational capacity of the node depends on its number of CPUs, amount of memory, basic storage space and other specifications. In other words, each node has its own processing speed, which can be expressed in number of Cycles per Unit Time (CPUT).

Jobs and operations: A job is considered as a single set of multiple atomic operations/tasks. Each operation is typically allocated to execute on one single node without pre-emption. It has input and output data and processing requirements in order to complete its task. The operation has a processing length expressed in number of cycles.

Task scheduling: A task scheduling is the mapping of tasks to a selected group of resources which may be

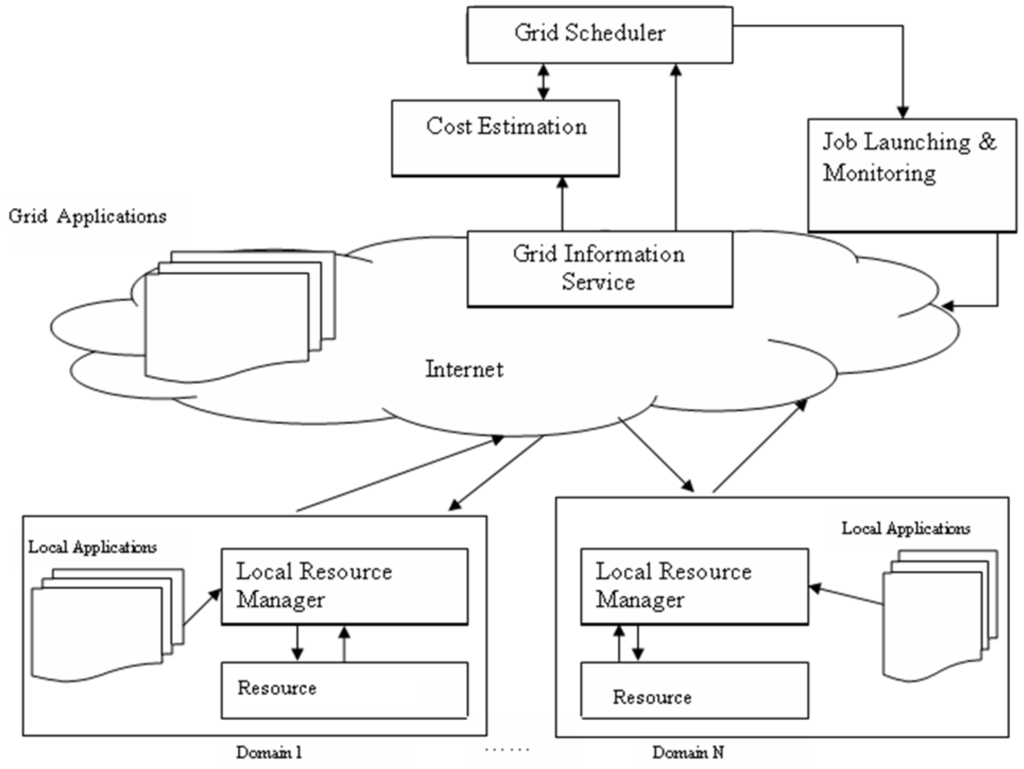

Fig. 1: A logical grid scheduling architecture

distributed in multiple administrative domains. A scheduling problem is specified by a set of machines, a set of jobs/operations, optionality criteria, environmental specifications and by other constraints.

The grid scheduling process and components: The grid scheduling process can be generalized into three stages: resource discovering and filtering, resource selecting and scheduling according to certain objectives and job submission (Schopf, 2001). Figure 1 depicts a model of grid scheduling system in which functional components are connected by two types of data flow: resource or application information flow and task or task scheduling command flow.

A Grid Scheduler (GS) receives applications from grid users, selects feasible resources for these applications according to acquired information from the Grid Information Service (GIS) module and finally generates application-to-resource mappings, based on certain objective functions and predicted resource performance. Grid schedulers usually cannot control grid resources directly. But they work like brokers or agents (Berman *et al*., 2003), or even tightly coupled with the applications as the application-level scheduling scheme proposes (Berman *et al*., 1996; Sun and Ming, 2003). They are not necessarily located in the same domain with the resources which are visible to them. Figure 1 shows only one grid scheduler, but in reality multiple schedulers might be deployed and organized to form different structures (centralized, hierarchical and decentralized (Hamscher *et al*., 2000) according to different concern, such as performance or scalability. Grid level scheduler is referred as Meta scheduler in the literature (Mateescu, 2003) and which is not an indispensible component in the Grid infrastructure.

The role of GIS is to provide information about the status of available resources to grid schedulers. GIS is responsible for collecting and predicting the resource state information, such as CPU capacity, memory size, network bandwidth, software availabilities and load of a site in a particular period. Application Profiling (AP) is used to extract properties of applications. Analogical Bench marking (AB) provides a measure of how well a resource can perform a given type of job (Khokhar *et al*., 1993; Siegel *et al*., 1996). On the basis of knowledge from AP and AB and following a certain performance model (Berman, 1998), cost estimation computes the cost of candidate schedules, from which the scheduler chooses those that can optimize the objective functions.

The Launching and Monitoring (LM) module is known as the ‘binder’ which implements a finally determined schedule by submitting applications to selected resources, staging input data and executables and monitoring the execution of the applications.

A Local Resource Manager (LRM) is mainly responsible for local scheduling inside a resource domain and reporting resource information to GIS.

Scheduling problem formulation: The objective of the proposed job scheduling algorithm is to minimize the completion time and to utilize the nodes effectively. Any job J_j has to be processed in one of the Grid nodes R_i until completion. Since all nodes at each stage are identical and pre-emption is not allowed, to define a schedule it suffices to specify the completion time for all tasks.

The grid job scheduling problem consists of scheduling m jobs with given processing time on n resources. Let J_j be the independent user jobs, $j = \{1, 2, 3, \dots m\}$. Let R_i be the heterogeneous grid nodes, $i = \{1, 2, 3, .., n\}$. The speed of each resource is expressed in number of Cycles per Unit Time (CPUT). The length of each job is expressed in number of cycles. The information related to job length and speed of the resource is assumed to be known based on user supplied information, experimental data and application profiling or other techniques (Garg *et al.*, 2010).

Let C_{ij} ($i \in \{1, 2, \dots .n\}, j \in \{1, 2, \dots .m\}$) be the completion time that the grid node R_i finishes the job J_j, $\sum C_i$ represents the time that the resource R_i finishes all the jobs scheduled for itself. Makespan is a measure of the throughput of the heterogeneous computing system. Makespan is defined as:

$$makespan, C_{max} = \max\left(\sum C_i\right) \tag{1}$$

For example thirteen jobs with job length 6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52 and 60 number of cycles, respectively, which are allocated to three grid nodes with speed 5, 3 and 2 CPUT, respectively. The completion time of a particular job is its job length divided by the speed of the grid node for which it has been allocated.

The schedule solution with the mapping of jobs with grid nodes is as follows.

J_3, J_4, J_5, J_9, J_{11} and J_{13}, respectively are mapped to grid node 1. J_7, J_{10} and J_{12}, respectively are mapped to grid node 2.

J_1, J_2, J_6 and J_8, respectively are mapped to grid node 3. The completion time for individual node is given below:

$$\sum C_1 = C_{1,3} + C_{1,4} + C_{1,5} + C_{1,9} + C_{1,11} + C_{1,13} = 41.6$$
$$\sum C_2 = C_{2,7} + C_{2,10} + C_{2,12} = 41.3333$$
$$\sum C_3 = C_{3,1} + C_{3,2} + C_{3,6} + C_{3,8} = 41$$
$$Makespan = C_{max} = max\left(\sum C_i\right) = 41.6$$

BBO ALGORITHM FOR SCHEDULING JOBS ON COMPUTATIONAL GRID

This section describes the biogeography based optimization technique and the different steps involved therein. Methodology of application of BBO technique to different cases of Grid Job Scheduling problem has also been presented in this section.

BBO algorithm: Biogeography describes how species migrate from one island to another, how new species arise and how species become extinct. An island is any habitat that is geographically isolated from other habitats. Geographical areas that are well suited as residences for biological species are said to have a high Habitat Suitability Index (HSI). The variables that characterize habitability are called Suitability Index Variables (SIVs). Habitats with a high HSI tend to have large number of species, while those with a low HSI have a small number of species. Habitats with a high HSI have many species that migrate to nearby habitats, simply by virtue of the large number of species that they host. Migration of some species from one habitat to other habitat is known as emigration process. When some species enter into one habitat from any other outside habitat, it is known as immigration process. Habitats with a high HSI have a low species immigration rate because they are already nearly saturated with species. By the same token, high HSI habitats have a high emigration rate. Habitats with a low HSI have a high species immigration rate because of their sparse populations. This immigration of new species to low HSI habitats may raise the HSI of that habitat, because the suitability of a habitat is proportional to its biological diversity. In BBO, each individual has its own immigration rate λ and emigration rate μ. A good solution has higher μ and lower λ, vice versa. The immigration rate and the emigration rate are functions of the number of species in the habitat. They can be calculated using (2) and (3):

$$\mu_k = \frac{Ek}{n} \tag{2}$$

$$\lambda_k = I\left(1 - \frac{k}{n}\right) \tag{3}$$

where, I is the maximum possible immigration rate, E is the maximum possible emigration rate, k is the number of species of the kth individual and n is the maximum number of species. Equation (2) and (3) are just one method for calculating λ and μ. There are other different options to assign them based on different specie models (Simon, 2008).

BBO concept is based on the two major steps, migration and mutation. Mathematically the concept of emigration and immigration can be represented by a probabilistic model. Let us consider the probability P_s that the habitat contains exactly S species at time t. P_s changes from time t to time t+Δt as follows:

$$P_s(t + \Delta t) = P_s(t)(1 - \lambda_s \Delta t - \mu_s \Delta t) + P_s - 1\lambda_s - 1\Delta t + P_s + 1\mu_s + 1\Delta t \tag{4}$$

$$\dot{P}_s = \begin{cases} -(\lambda_s + \mu_s)P_s + \mu_{s+1} P_{s+1} & S = 0 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1} P_{s-1} + \mu_{s+1} P_{s+1} & 1 \le S \le S_{max} - 1 \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1} P_{s-1} & S = S_{max} \end{cases} \tag{5}$$

where, λ_s and μ_s are the immigration and emigration rates when there are S species in the habitat. If time Δt is small enough so that the probability of more than one immigration or emigration can be ignored then taking the limit of (4) as $\Delta t \rightarrow 0$ gives (5).

Mutation rate of each set of solution can be calculated in terms of species count probability using the Eq. (6):

$$m(S) = m_{\max}\left(\frac{1-P_s}{P_{\max}}\right) \quad (6)$$

where, m_{max} is the maximum mutation rate and $P_{\max}$ is the maximum probability. The pseudo code for BBO algorithm is illustrated in Algorithm 1.

Solution representation: The solution for the job scheduling problem is used to represent individual habitat. The complete habitat set (population) with size N is represented in (7):

$$H = \left[H^1 \quad H^2 \quad H^3 \quad .... \quad H^i \quad .... \quad H^N \right] \quad (7)$$

$$H = \begin{bmatrix} SIV^{11} & SIV^{12} & SIV^{13} & ... & SIV^{1j} & ... & SIV^{1n} \\ SIV^{21} & SIV^{22} & SIV^{23} & ... & SIV^{2j} & ... & SIV^{2n} \\ SIV^{31} & SIV^{32} & SIV^{33} & ... & SIV^{3j} & ... & SIV^{3n} \\ SIV^{41} & SIV^{42} & SIV^{43} & ... & SIV^{4j} & ... & SIV^{4n} \\ ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... \\ SIV^{N1} & SIV^{N2} & SIV^{N3} & ... & SIV^{Nj} & ... & SIV^{Nn} \end{bmatrix} \quad (8)$$

Algorithm 1 biogeography based optimization algorithm:

```
Initialize the BBO parameters
Create a random set of habitats (population) $H_1$, $H_2$, .., $H_n$.
Compute HSI values;
while the halting criterion is not satisfied do
  Compute immigration rate λ and emigration rate μ for each
    habitat based on HSI;
    for each habitat (solution)
      for each SIV (solution feature)
      /* Migration process */
          Select habitat $H_i$ with probability α $\lambda_i$
          if $H_i$ is selected then
              Select $H_j$ with probability α $\mu_j$
          if $H_j$ is selected then
              $H_i$ (SIV) ← $H_j$ (SIV)
          end if
        end if
      /* Mutation process */
      Select $H_i$ (SIV) based on mutation probability $m_i$;
          if $H_i$ (SIV) is selected then
            Replace $H_i$ (SIV) with a randomly generated SIV;
          end if
        next for
          Recompute HSI values;
    next for
end while
```

Table 1: Representation of solution, (a) job-to-resource representation for the grid job scheduling problem, (b) mapping of jobs with grid resource

(a)

2	1	2	3	1	2	3	1	2	3	2	1	1

(b)

Grid resource 1	J2	J5	J8	J12	J13
Grid resource 2	J1	J3	J6	J9	J11
Grid resource 3	J4	J7	J10		

where i = 1, 2, ….. N, j = 1, 2, 3, ……n and $1 \leq SIV \leq m$ (n is the number of jobs, m is the number of Grid nodes and SIV is the Suitability Index Variable). Here H_i is the position vector of the habitat i. Each habitat (chromosome) is one of the possible solutions for the job scheduling problem. The element H_{ij} of H_i is the jth position component of habitat i or in other words H_{ij} is the jth SIV of the ith habitat. All SIVs in each habitat are represented as integers.

The solution is represented as an array of length equal to the number of jobs. The value corresponding to each position j in the array represents the SIV in a habitat which is actually the resource to which job j was allocated. Table 1 illustrates the representation of the SIV for the job resource pair (13, 3). The first element of the array denotes the first job (J_1) which is assigned to the Grid resource 2; the second element denotes the second job (J_2) which is assigned to the Grid resource 1 and so on.

Initialization of SIV: Each element of the habitat matrix, i.e., each SIV of a given habitat set H, is initialized randomly with the value satisfying the resources limit R_j, where, j = 1, 2,… m (Table 1).

Now the steps of algorithm to solve grid job scheduling are given below:

1. For initialization, choose the number of jobs i.e., number of SIV is n, size of habitat is N. Initialize total number of resources and jobs. Also initialize the BBO parameters like habitat modification probability P_{mod}, mutation probability, maximum mutation rate m_{max}, Maximum immigration rate I, Maximum emigration rate E, lower and upper bound for immigration probability per gene, λ_{lower} and λ_{upper}, step size for numerical integration dt, elitism parameter P, etc. Set maximum number of iteration.
2. Each SIV of a given habitat of H matrix is initialized using the concept mentioned in

"Initialization of *SIV*". Each habitat represents a potential solution to the given problem.

3. Calculate HSI for each habitat set of the total habitat set for given emigration rate μ and immigration rate λ using Eq. (2) and (3).
4. Based on the makespan value, elite habitats are identified. Here elite terms are used to indicate those habitat sets which give minimum fitness value. Top "*P*" habitat sets are kept as it is after individual iteration without making any modification on it. Valid species *S* in grid job scheduling problem is identified by considering finite fitness values of habitats.
5. Perform migration operation on those *SIV*s of each non-elite habitat, selected for migration. Algorithm 2 describes the migration operation of BBO based grid job scheduling problem.
6. For each habitat, update the probability of its species count using Eq. (5). Then, mutate each non-elite habitat based on its probability using Eq. (6) and recalculate the fitness values.
7. Go to step 4 for the next iteration. This loop can be terminated after a predefined number of iterations.

Algorithm 2 Habitat migration for grid job scheduling:

```
/* To calculate species count */
for  i = 1 to N
        if  fitness of habitat set i< ∞
            Speciescount of habitat i = N-i;
        else
            Speciescount of habitat i = 0;
        end if
end for
/* calculate value of λ and μ for each habitat set */
for i = 1 to N
        λ (i) = I * (1-Speciescount of habitat i/N);
        μ (i) = E * (Speciescount of habitat i/N);
end for
λmin = min (λ);  λmax = max (λ);
/ * To select habitat and SIV for generating new habitat
after migration */
for k = 1 to N
   if a randomly generated number<Pmod
        /* To normalize the immigration rate */
   λscale  =  λlower + (λupper - λlower) * (λ (k) -λmin)
/ (λmax - λmin)
        /* To pick up a habitat from which to obtain a
        feature */
        for  j = 1 to n
          if a randomly generated number<λscale
              RandomNum = rand * sum (μ);
              Select = μ (1);
              SelectIndex = 1;
                  while (RandomNum>Select) and
                  (SelectIndex<N)
                      SelectIndex = SelectIndex+1;
                        Select = Select + μ (SelectIndex);
                  endwhile
                  Newly generated habitat (k, j) = Old habitat
                  (Selectindex, j);
                  /* To check the feasibility of new habitat */
                   for z = 1 to n
                     if Newly generated  habitat  (k, z) ≠ Ri,
                     (i = 1, 2, … m)
                    Repeat the procedure for generating new
                    habitat
                           end if
                   end for
              else
                 Newly generated habitat (k, j) = Old habitat
                 (k. j);
              end if
          end for
        end if
end for
```

SIMULATION ON BBO BASED GRID JOB SCHEDULING ALGORITHM

Proposed BBO algorithm has been applied to grid job scheduling problem in four different test cases for verifying its feasibility. These are a (3, 13) -resource job pair, a (5, 100) -resource job pair, a (8, 60) -resource job pair and a (10, 50) -resource job pair. The numerical simulations are carried out with the dataset used and tested in the study (Liu *et al*., 2010).The performance of the proposed algorithm is compared with ACO, PSO, DE and GA. Specific parameter settings of all the considered algorithms after performing the extensive experiments are described in Table 2. The adopted procedure for the determination of the parameters of the proposed algorithm is detailed in this study.

Table 2: Parameter settings for the algorithms

Algorithm	Parameter name	Parameter value
GA	Size of the population	125
	Probability of crossover	1
	Probability of mutation	0.100
PSO	Size of the population	125
	Particle swarm neighbourhood size	0
	Inertial constant	0.800
	Social coefficient	1.490
	Inertia weight	1.490
BBO	Habitat size	125
	habitat modification probability	1.000
	Immigration probability bounds per gene	[0, 1]
	Step size for numerical integration	0.200
	Maximum immigration and emigration rate for each island	1.000
	Mutation probability	0.005
DE	Size of the population	125
	Cross over factor	0.500
	Scaling factor	0.500
ACO	Size of the population	125
	Pheromone update constant	20
	Exploration constant	1
	Pheromone sensitivity	1
	Visibility sensitivity	5

Table 3: Influence of parameters on BBO performance for job scheduling problem (5, 100)

		With mutation											
		0.005			0.05			0.5			Without mutation		
Case	dt	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
1	0.1	85.447	85.549	85.480	85.463	85.921	85.523	85.454	85.683	85.511	85.450	85.736	85.521
2	0.2	85.443	85.543	85.410	85.472	85.893	85.536	85.450	85.643	85.508	85.453	85.986	85.745
3	0.4	85.449	85.765	85.537	85.454	85.562	85.505	85.448	85.651	85.513	85.469	85.741	85.652
4	0.5	85.463	85.603	85.540	85.457	85.573	85.504	85.467	85.643	85.512	85.456	85.741	85.642
5	0.8	85.464	85.545	85.514	85.448	85.603	85.511	85.448	85.621	85.509	85.468	85.663	85.556
6	1.0	85.448	85.643	85.682	85.450	85.582	85.504	85.460	85.730	85.528	85.445	85.718	85.657
7	1.2	85.454	85.822	85.520	85.468	85.637	85.522	85.452	85.645	85.498	85.476	85.685	85.533
8	1.3	85.456	85.958	85.738	85.451	85.711	85.516	85.465	85.621	85.518	85.454	85.759	85.550
9	1.5	85.458	85.582	85.499	85.453	85.676	85.520	85.456	85.579	85.503	85.455	85.677	85.613
10	2.0	85.447	85.615	85.512	85.446	85.711	85.499	85.499	85.754	85.618	85.465	85.813	85.673

Min.: Minimum; Max.: Maximum; Avg.: Average

Table 4: Effect of habitat size on results of job scheduling problem (5, 100)

Habitat size	No. of hits to 85.4000 - 85.5999	Min. makespan	Max. makespan	Avg. makespan
20	13	85.467	85.572	85.576
50	14	85.443	85.543	85.420
75	16	85.500	85.722	85.610
100	14	85.504	85.707	85.626
125	20	85.440	85.540	85.413
150	12	85.520	86.053	85.650

Min.: Minimum; Max.: Maximum; Avg.: Average

Determination of parameters for BBO algorithm: To get optimal solution using the BBO algorithm, the suitable value of the parameters like mutation probability, step of integration *dt* and habitat size N have to be determined. As the job scheduling problem (5.100) is a large scale problem, (5.100) has been chosen to determine the various parameters of BBO algorithm. To find optimum values for "step size of integration *dt*" and "mutation probability", the following procedures have been adopted:

- The habitat size is fixed at 50.
- Step of integration is increased from 0.1 to 2 in suitable steps as shown in Table 1 and mutation probability is changed to three different values of 0.005, 0.05 and 0.5, respectively. Performance of BBO algorithm in grid job scheduling system for the resource job pair (5, 100) is calculated for all the above mentioned combinations. For each combination, 50 independent trails have been made with 500 iterations per trail.
- Step of integration is again increased from 0.1 to 2 in same steps for the same problem as mentioned above and mutation probability is not considered in this case.
- In case of BBO algorithm, based on simulation results obtained for different combination of parameters given in Table 3, step size of integration dt 0.2 and mutation probability 0.005 gave better makespan more consistently. The obtained minimum makespan 85.443 is also less when compared to the remaining cases.

Effect of habitat size and maximum number of iteration on BBO algorithm: Change in habitat size affects the performance of BBO algorithm. Large or a small habitat size may not be capable of searching for the minimum, particularly in complex multimodal problems. The optimum habitat size is found to be related to the problem dimension and complexity. Table 4 shows the performance of the BBO algorithm for different habitat size of 20, 50, 75, 100, 125 and 150, respectively for the resource job pair (5, 100) of Grid job scheduling system.

A habitat size of 125 resulted in achieving global solutions more consistently for the test system. From Table 4, it is found that the habitat size 125 recorded the best result compared with other habitat sizes. Increasing the habitat size beyond this value did not produce any significant improvement; rather it increases the simulation time which is not desirable in real-time problems.

The nature of convergence of BBO algorithm had been observed for all kind of problems by few tests. After that, the number of iterations had been fixed as 100. In the proposed algorithm, the parameters are set as specified in Table 2.

Comparative study: All the jobs and the nodes were submitted at one time. Each experiment (for each algorithm) was repeated 25 times with different random seeds. Each trail had a fixed number of 100 iterations. The makespan values of the best solutions throughout the optimization run were recorded and the averages and the standard deviations were calculated from the 25 different trails. The grid scheduling algorithm should generate the schedules as fast as possible in a grid environment. So the completion time is used as one of the criteria for improving their performance.

Solution quality: Table 5 shows the performance comparison of BBO algorithm with ACO, PSO, GA and DE. Figure 2 illustrates the performance for the pair (3, 13). As the performance plot of PSO had many fluctuations, it average makespan and the standard deviation are recorded by both GA and BBO. It is noticed that all algorithms except PSO allocate the jobs evenly for all grid nodes from Fig. 3.

Table 5: Performance comparison between ACO, BBO, GA, DE and PSO

		Instance			
Algorithm	Item	(3, 13)	(5, 100)	(8, 60)	(10, 50)
ACO	Average makespan	41.68	104.08	67.67	51.10
	Standard deviation	±0.16	±5.71	±7.25	±2.36
	Time	3.96	7.90	11.83	10.02
BBO	Average makespan	41.64	85.47	43.01	37.46
	Standard deviation	±0.12	±0.02	±0.42	±0.68
	Time	2.73	8.73	6.16	5.40
DE	Average makespan	41.68	88.05	47.10	42.08
	Standard deviation	±0.16	±0.70	±1.18	±0.93
	Time	3.47	11.66	8.29	7.60
GA	Average makespan	41.64	86.02	43.23	37.90
	Standard deviation	±0.12	±0.23	±0.54	±0.68
	Time	3.70	9.41	7.09	6.50
PSO	Average makespan	49.06	105.76	49.95	49.75
	Standard deviation	±3.65	±3.26	±2.00	±1.67
	Time	5.17	10.79	8.30	7.89

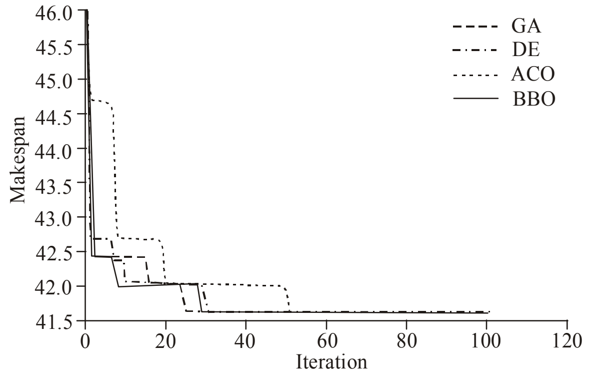

Fig. 2: Performance for job scheduling (3, 13)

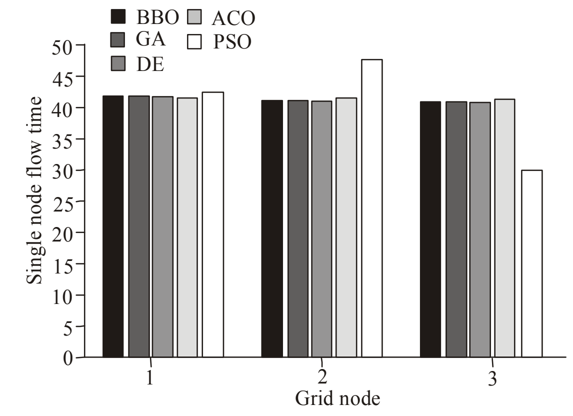

Fig. 3: Resource allocation for job scheduling (3, 13)

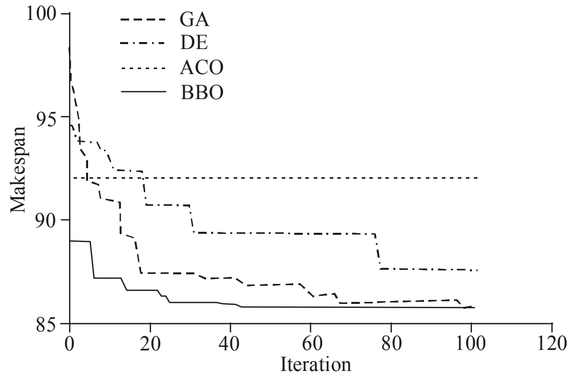

Fig. 4: Performance for job scheduling (5, 100)

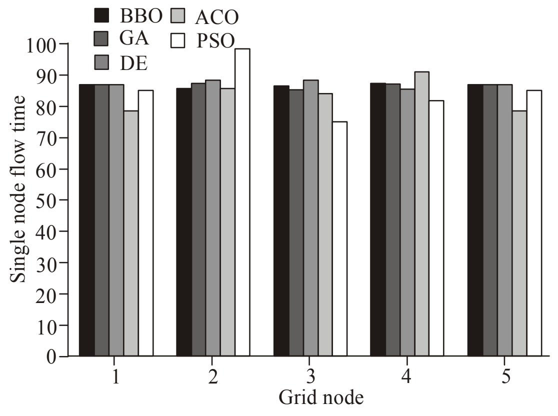

Fig. 5: Resource allocation for job scheduling (5, 100)

Figure 4 illustrates the performance for GA, ACO, BBO and DE algorithms during the search process for (5, 100). For this case, the BBO algorithm yields the minimum average makespan and the standard deviation.

Individual node flow time of best run is found to be {85.22, 84.76, 85.66, 85.67, 85.51} for BBO. GA produces the flowtime as {85.26, 85.55, 84.68, 85.76, 85.64}. The result for DE is {85.24, 86.73, 87.50, 84.97, 84.11}. ACO reports {76.71, 84.26, 83.15, 89.30, 91.93} and PSO yields {83.39, 96.8, 74.04, 79.87, 97.24}. It is realized that the performance of BBO and GA is good when compared with other algorithms from Fig. 5 and 6. The average makespan for (8, 60) is found to be 43.01 for BBO which is the minimum value as evident from the Table 5. Individual node flow time for this case is illustrated in Fig. 7. This reveals that BBO is better than other algorithms.

Figure 8 illustrates the performance for the pair (10, 50). The minimum average makespan is recorded by BBO as 37.46 and the standard deviation for BBO and GA is found to be the same for this case which is also the minimum value compared with others. It is noted that the resources are effectively utilized by BBO

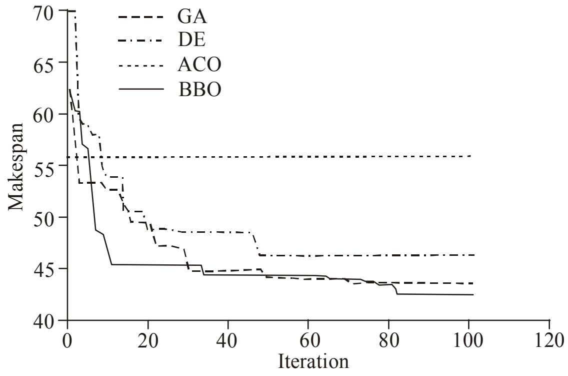

Fig. 6: Performance for job scheduling (8, 60)

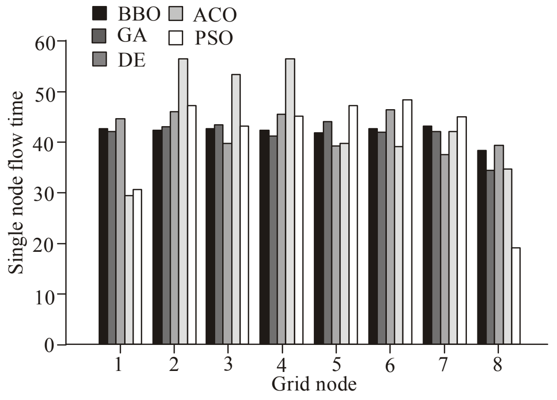

Fig. 7: Resource allocation for job scheduling (8, 60)

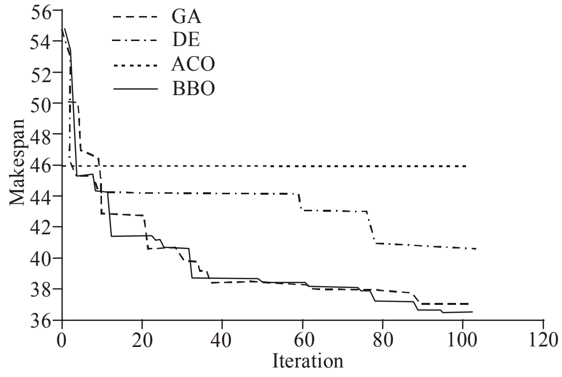

Fig. 8: Performance for job scheduling (10, 50)

algorithm than others when referring to Fig. 9. In general, it is found that the performance of BBO is better and competitive with GA while producing good quality schedules. Next DE places its remark and it is followed by PSO and ACO in producing quality schedules.

Computational efficiency: In general, for large (R, J) pairs, the completion time is comparatively larger. BBO usually spent the least time for allocating all the jobs on the grid node, GA was the second. For large resource job pair, the completion time of PSO is competitive with DE as referred from Table 5. ACO had to spend more time to complete the scheduling. It is noted that

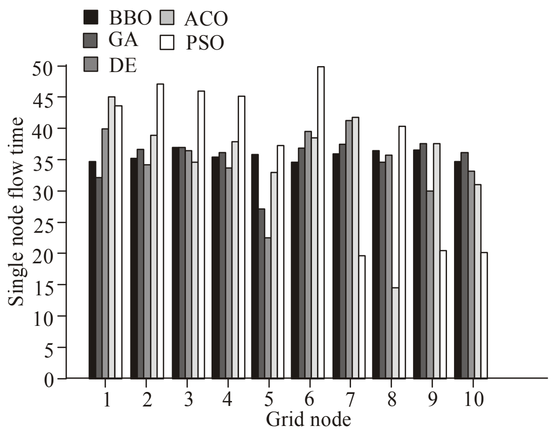

Fig. 9: Resource allocation for job scheduling (10, 50)

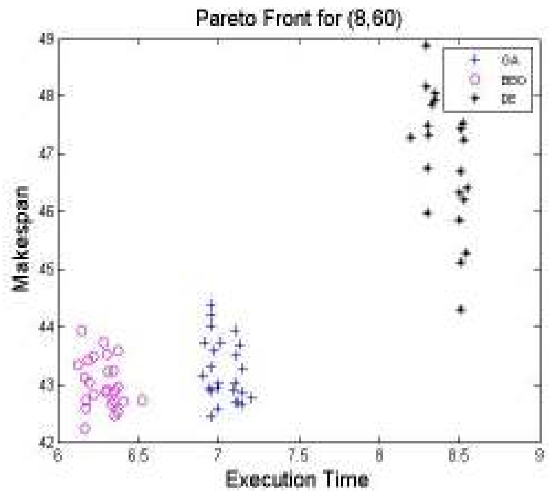

Fig. 10: Pareto front obtained by three algorithms for (8, 60)

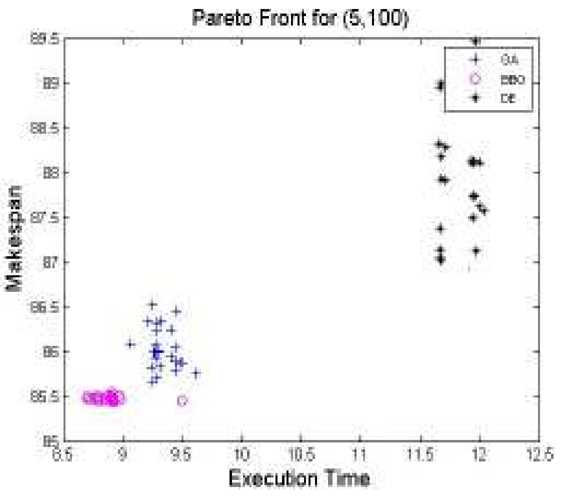

Fig. 11: Pareto front obtained by three algorithms for (5, 100)

BBO usually spends the shortest time to accomplish the various job scheduling tasks and produces the best results among all the five algorithms considered.

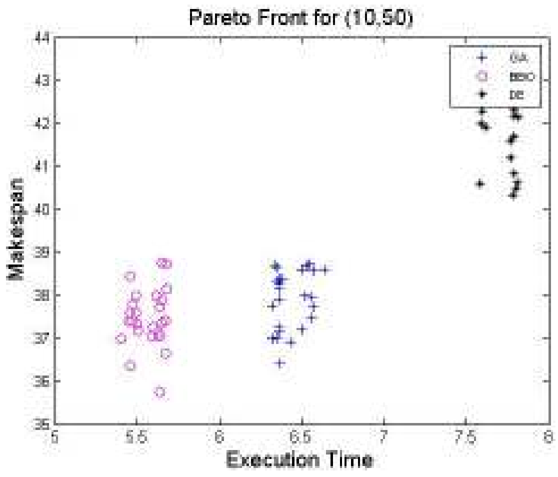

Fig. 12: Pareto front obtained by three algorithms for (10, 50)

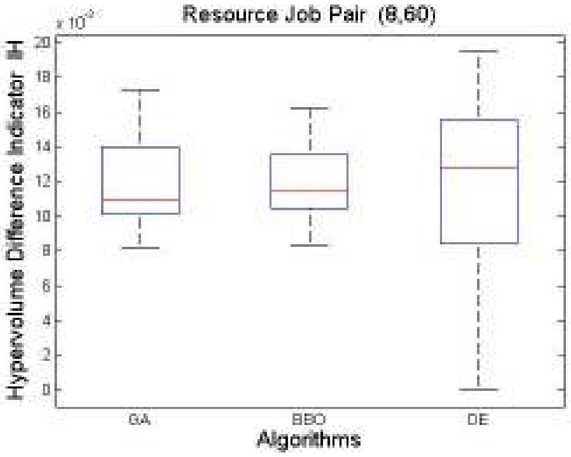

Fig. 13: I_H^- measure of three algorithms for (8, 60)

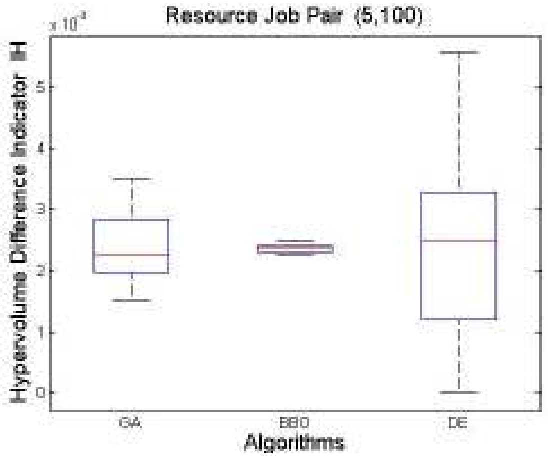

Fig. 14: I_H^- measure of three algorithms for (5, 100)

Performance assessment: From Table 5, it is found that BBO gives the best makespan value for all cases, then GA follows next and then DE in giving better results. Hence GA and DE are taken into account to

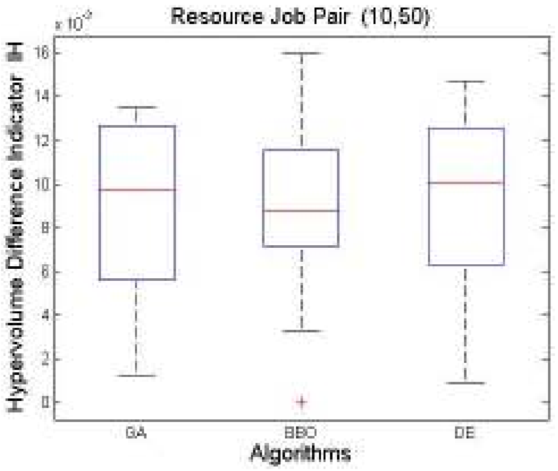

Fig. 15: I_H^- measure of three algorithms for (10, 50)

assess the performance with BBO. In order to compare the performance of the proposed algorithm with other scheduling algorithms, it is necessary to examine the extent of minimization of the obtained non-dominated solutions produced by each algorithm for the considered objective and the spread of their solutions. Figure 10 to 12 show the non-dominated solutions obtained at the end of simulation trial (average over 25 runs) for BBO, GA and DE algorithms for 3 different test cases.

For all problems, the solution obtained by BBO is better than solutions found by GA and DE. In order to present a comprehensive comparison of the overall quality of these alternative approaches, the experiment for each algorithm was repeated 25 times with different random seeds for each resource job pair. The reference set, R had been constructed by merging all of the archival non-dominated solutions found by each of the algorithms for a given resource job pair across 25 runs. Then the hyper volume difference indicator IH^- (Huband *et al*., 2006) had been used to measure the differences between non-dominated fronts generated by the algorithms and the reference set R.

The objective values are normalized to find the hyper volume difference indicator (Huang *et al*., 2007). IH^- measures the portion of the objective space that is dominated by R. The lower the value of IH^-, the better the algorithm performs. Box plots for different test cases clearly prove that BBO algorithm is better than GA and DE (Fig. 13 to 15).

From the simulation result of BBO algorithm in solving grid job scheduling problems, it is seen that the performance of BBO algorithm is much better than other optimization techniques mentioned in this study.

CONCLUSION

In this study, we analyzed the job scheduling problems on computational grids. For scheduling problems, we consider genetic algorithm, differential evolution algorithm, ant colony optimization, particle swarm optimization and BBO algorithm.

This study presents a novel grid job scheduling approach based on BBO algorithm to optimize the makepan and flowtime. The BBO algorithm has an ability to find better quality solution and has better convergence characteristics and computational efficiency. It is clear from the results obtained by different trials that the proposed BBO method has good convergence property and it avoids the shortcoming of premature convergence of other optimization techniques to obtain better quality solution. As the status of resource is dynamic within the grid environment, it is necessary to produce the faster and feasible schedules. Simulation results show that BBO is capable of generating the solution within a minimum period of time. The comparative study demonstrates the efficiency and effectiveness of the proposed approach and the IH^{-} indicator shows that BBO performs better than other algorithms. Hence BBO can be applied for grid job scheduling problems. In future work, we will develop adaptive BBO algorithm for multi-objective complex scheduling problems and stochastic scheduling problems.

REFERENCES

Berman, F., 1998. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, Springer-Verlag, San Mateo, CA.

Berman, F., R. Wolski, S. Figueria, J. Schopf and G. Shao, 1996. Application-level scheduling on distributed heterogeneous networks. Proceeding of the 1996 ACM/IEEE Conference on Supercomputing, Article No: 39.

Berman, F., R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su and D. Zagorodnov, 2003. Adaptive computing on the grid using apples. IEEE T. Parall. Distr., 14(4): 369-382.

Bhattacharya, A. and P.K. Chattopadhyay, 2010. Biogeography-based optimization for different economic load dispatch problems. IEEE T. Power Syst., 25(2): 1064-1077.

Braun, T.D., H.J. Siegel, N. Beck, D.A. Hensgen and R.F. Freund, 2001. A comparison of eleven static heuristics for mapping a class of independent tasks on heterogeneous distributed system. J. Parallel Distr. Com., 61(6): 810-837.

Dong, F. and S.G. Akl, 2006. Scheduling algorithms for grid computing: State of the art and open problems. Technical Report No. 2006-504. School of Computing, Queen's University Kingston, Ontario.

Du, D., D. Simon and M. Ergezer, 2009. Biogeography-based optimization combined with evolutionary strategy and immigration refusal. Proceeding of the IEEE Conference on Systems, Man and Cybernetics. San Antonio, Texas, pp: 997-1002.

Foster, I. and A. Iamnitchi, 2003. On death, taxes and the convergence of peer-to-peer and grid computing. Proceeding of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, USA.

Foster, I. and C. Kesselmann, 2004. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, USA.

Foster, I., C. Kesselman and S. Tuecke, 2001. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. Supercomput. Ap., 15(3): 200-220.

Garg, S.K., R. Buyya and H.J. Siegel., 2010. Time and cost trade-off management for scheduling parallel applications on utility grids. Future Gener. Comp. Sy., 26(8): 1344-1355.

Hamscher, V., U. Schwiegelshohn, A. Streit and R. Yahyapour, 2000. Evaluation of job-scheduling strategies for grid computing. Proceeding of 1st IEEE/ACM International Workshop on Grid Computing (GRID'00). Bangalore, India, pp: 191-202.

Huang, V.L., A.K. Qin, K. Deb, E. Zitzler, P.N. Suganthan, J.J. Liang, M. Preuss and S. Huband, 2007. Problem definitions for performance assessment on multi-objective optimization algorithms. Technical Report, Nanyang Technological University, Singapore.

Huband, S., P. Hingston, L. Barone and L. While, 2006. A review of multiobjective test problems and a scalable test problem toolkit. IEEE T. Evolut. Comput., 10(5): 477-506.

Ibarra, O.H. and C.E. Ki, 1977. Heuristic algorithms for scheduling independent tasks on nonidentical processors. JACM, 24(2): 280-289.

Jarvis, S.A., D.P. Spooner, H.N. Lim Choi Keung, G.R. Nudd, J. Cao and S. Saini, 2003. Performance prediction and its use in parallel and distributed computing systems. Proceeding of the IEEE/ACM International Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems. Nice, France.

Khokhar, A.A., V.K. Prasanna, M.E. Shaaban and C.L. Wang, 1993. Heterogeneous computing: Challenges and opportunities. IEEE Comput., 26(6): 18-27.

Krauter, K., R. Buyya and M. Maheswaran, 2002. A taxonomy and survey of grid resource management systems for distributed computing. Software Pract. Exper., 32: 135-164.

Liu, H., A. Abraham and A.E. Hassanien, 2010. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Gener. Comp. Sy., 26(8): 1336-1343.

Lohokare, M.R., S.S. Pattnaik, S. Devi, K.M. Bakwad and J.G. Joshi, 2009. Parameter calculation of rectangular microstrip antenna using biogeography-based optimization. Proceeding of Applied Electromagnetics Conference (AEMC). Kolkata, DOI: 10.1109/AEMC.2009.5430676.

Lozovyy, P., G. Thomas and D. Simon, 2011. Biogeography-based optimization for robot controller tuning. Computational Modeling and Simulation of Intellect: Current State and Future Perspectives. IGI Global Publication, Chapter 7, pp: 162-181.

Ma, H., S. Ni and M. Sun, 2009. Equilibrium species counts and migration model tradeoffs for biogeography-based optimization. Proceeding of the IEEE Conference on Decision and Control. Shanghai, P.R. China, pp: 3306-3310.

Mateescu, G., 2003. Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation. Int. J. High Perform. C., 17(3): 209-218.

Panchal, V., P. Singh, N. Kaur and H. Kundra, 2009. Biogeography based satellite image classification. Int. J. Comput. Sci. Inform. Secur., 6(2): 269-274.

Schopf, J., 2001. Ten Actions When Super Scheduling, document of Scheduling Working Group, Global Grid Forum. Retrieved form: http://www.ggf.org/documents/GFD.4.pdf, July 2001.

Siegel, H.J., H.G. Dietz and J.K. Antonio, 1996. Software support for heterogeneous computing. ACM Comput. Surv., 28(1): 237-239.

Simon, D., 2008. Biogeography-based optimization. IEEE T. Evolut. Comput., 12(6): 702-713.

Singh, U., H. Singla and T. Kamal, 2010. Design of Yagi-Uda antenna using biogeography based optimization. IEEE T. Antenn. Propag., 58(10): 3375-3379.

Song, Y., M. Liu and Z. Wang, 2010. Biogeography-based optimization for the traveling salesman problems. Proceeding of the 3rd International Joint Conference on Computational Science and Optimization (CSO, 2010). Huangshan, Anhui, China, pp: 295-299.

Sun, X.H. and W. Ming, 2003. Grid harvest service: A system for long-term, application-level task scheduling. Proceeding of 2003 International Parallel and Distributed Processing Symposium, ISSN: 1530-2075.