Future Computing and Informatics Journal

Volume 3
Issue 2 (2018) *Vol 3,Issue 2,2018*

Article 8

2018

Task schedul ing for cloud computing using multi-objective hybrid bacteria foraging algorithm

Sobhanayak Srichandan
Department of Computer Science, IIIT Bhubaneswar, Odisha, India, srichandan@iiit-bh.ac.in

Turuk Ashok Kumar
Department of Computer Science, NIT Rourkela, Odisha, India, akturuk@nitrkl.ac.in

Sahoo Bibhudatta
Department of Computer Science, NIT Rourkela, Odisha, India, bdsahu@nitrkl.ac.in

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 3 (2018) 210–230
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm

Sobhanayak Srichandan [a,*], Turuk Ashok Kumar [b], Sahoo Bibhudatta [b]

[a] Department of Computer Science, IIIT Bhubaneswar, Odisha, India
[b] Department of Computer Science, NIT Rourkela, Odisha, India

Received 10 October 2017; revised 19 January 2018; accepted 7 March 2018
Available online 5 June 2018

Abstract

Cloud computing is the delivery of computing services over the internet. Cloud services allow individuals and other businesses organization to use data that are managed by third parties or another person at remote locations. Most Cloud providers support services under constraints of Service Level Agreement (SLA) definitions. The SLAs are composed of different quality of service (QoS) rules promised by the provider. A cloud environment can be classified into two types: computing clouds and data clouds. In computing cloud, task scheduling plays a vital role in maintaining the quality of service and SLA. Efficient task scheduling is one of the major steps for effectively harnessing the potential of cloud computing. This paper explores the task scheduling algorithm using a hybrid approach, which combines desirable characteristics of two of the most widely used biologically-inspired heuristic algorithms, the genetic algorithms (GAs) and the bacterial foraging (BF) algorithms in the computing cloud. The main contributions of this article are twofold. First, the scheduling algorithm minimizes the makespan and second; it reduces the energy consumption, both economic and ecological perspectives. Experimental results show that the performance of the proposed algorithm outperforms than those of other algorithms regarding convergence, stability, and solution diversity.

Keywords: Cloud computing; Resource allocation; Pareto solutions; Bacteria foraging; Genetic algorithms

1. Introduction

With the ubiquitous growth of Internet access and big data in their volume, velocity, and variety through the Internet, cloud computing becomes more and more proliferating in the industry, academia, and society. Cloud computing is composed of distributed computing, grid computing, utility computing, and autonomic computing [1]. Cloud computing provides on-demand computing and storage services with high performance and high scalability. Several computing paradigms have promised to deliver this utility computing. Cloud computing is one such reliable computing paradigm. However, the rising energy consumption of cloud data centers has become a prominent problem. Task scheduling is an important step to improve the overall performance of the cloud computing. Traditional monitoring and management mechanisms are designed for enterprise environments, especially a unified environment. However, the large scale, heterogeneous resource provisioning places serious challenges for the management and monitoring mechanism in multiple data centers. To the best of our knowledge, this is the first paper to address the scheduling issue using multi-objective hybrid bacteria foraging algorithm in the IaaS cloud with the heterogeneous system.

In recent years, the problem of task scheduling on a distributed environment has caught the attention of researchers. Task scheduling is considered a critical issue in the

* Corresponding author.
E-mail addresses: srichandan@iiit-bh.ac.in (S. Srichandan), akturuk@nitrkl.ac.in (T. Ashok Kumar), bdsahu@nitrkl.ac.in (S. Bibhudatta).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

https://doi.org/10.1016/j.fcij.2018.03.004

Cloud computing environment by considering different factors like completion time, the total cost for executing all users' tasks, utilization of the resource, power consumption, and fault tolerance. The problem of finding the right compromise between the resolution time and the energy consumed by a precedence-constrained parallel application is a bi-objective optimization problem. The solution to this problem is a set of Pareto points. Pareto solutions are those for which improvement in one objective can only occur with the worsening of at least one other objective. Thus, instead of a unique solution to the problem, the solution to a bi-objective problem is a (possibly infinite) set of Pareto points. Task scheduling has been proved as an NP-complete problem [2,3]. Cloud computing not just help a decent variety of uses, yet in addition give a virtualized condition for the applications to keep running in an efficient and minimal effort way [4].

A cloud data center usually consists of a large group of servers connected to the Internet. A task scheduler is needed in a cloud data center to arrange task executions. The task scheduler has to efficiently utilize the resources of the cloud data center to execute tasks. The performance issues of the scheduling algorithm include the makespan and energy consumption. A good scheduler can use fewer resources and times to accomplish tasks execution. Using fewer resources implies that less energy is consumed. The minimization of energy consumption and makespan is one of the major issues for building large-scale clouds.

There are different prospects of cloud computing has been studied to exploit the diversity of it viz. designing and implementing scheduling strategies and algorithms for specific tasks fault-tolerant tasks with real-time deadlines or energy efficient tasks such as dependent or independent. There is certain inherent problem associated with resource provisioning and task scheduling. The optimization goals, once set at the design time, will be statically built into the task scheduling and resource provisioning algorithm and implementation as the monolithic system component, thus lacking flexibility and adaptability in the presence of changing workload characterization, resource provisioning and cloud execution environment. Lots of task scheduling and resource provisioning strategy and algorithms, though intended with varied different optimization objectives, often contribute to some widespread functional mechanism and employ comparable software engineering framework for execution. However, adding new scheduling competence needs to be done for each scheduling algorithm one at a time, which is not only monotonous but also costly and leads to error.

Natural selection tends to eliminate animals with poor foraging strategies through methods for locating, handling, and ingesting food and favors the propagation of genes of those animals that have successful foraging strategies, since they are more likely to obtain reproductive success. After many generations, poor foraging strategies are either eliminated or re-structured into good ones. Since a foraging organism/animal takes actions to maximize the energy utilized per unit time spent foraging, considering all the constraints presented by its own physiology, such as sensing and cognitive capabilities and environmental parameters (e.g., density of prey, risks from predators, physical characteristics of the search area), natural evolution could lead to optimization. It is essentially this idea that could be applied to complex optimization problems. The optimization problem search space could be modeled as a social foraging environment where groups of parameters communicate cooperatively for finding solutions to difficult engineering problems [5].

Authors in Ref. [6] has provided an elaborate idea about GA by introducing several variants for task scheduling in the Cloud computing environment. He has introduced an algorithm to solve task scheduling problem by modifying GA in which the initial population is generated by Max−Min approach to get more optimum results in term of "makespan". Authors in Ref. [7] propose a resource scheduling algorithm considering the execution time of every distinct workload, but most importantly, the overall performance is also based on type of workload i.e. with different QoS requirements (heterogeneous workloads) and with similar QoS requirements (homogenous workloads).

To accomplish tasks execution in parallel following questions must be answered: (1) how to distribute resources to tasks; (2) in what order the clouds should execute tasks since tasks have data dependencies; (3) how to schedule overheads when physical machines (PM) set up, finish or switch tasks. Resource allocation and scheduling can solve these three problems. Resource allocation and task scheduling have been studied in high-performance computing [8] and embedded systems [9]. However, the autonomic feature and the resource heterogeneity within clouds [10] and the PM implementation require different algorithms for resource allocation and task scheduling in the IaaS cloud computing, especially in the federated, heterogeneous multi-cloud system.

Although a large number of articles have been published on analysis of the foraging behavior and self-adaptability properties of BFA as a single objective optimizer, till date, to the best of our knowledge, little such analysis exists for the multi-objective hybrid BFA algorithms. In this paper we propose a new multi-objective hybrid bacteria foraging algorithm (MHBFA) composed of genetic algorithm and a multi-objective bacteria foraging algorithm to find the set of Pareto-front solutions, hence called MHBFA. The major components of MHBFA algorithm are selection, mutation, and crossover from the genetic algorithm and multi objective optimization BFA algorithm [11−13], proposed by Passino in 2002, is a new comer to the family of nature-inspired optimization algorithms. BFA is a relatively new swarm intelligence algorithm inspired by the foraging behavior of *Escherichia coli* (*E. coli*) in human intestines. There exists a unique communication mechanism between individuals of *E. coli* for the communication purposes. We consider combinatorial optimization problem to formulate the proposed the task scheduling problem for minimization of energy and makespan.

The major contributions of this paper are:

- We propose a scheduling algorithm for heterogeneous cloud environment.
- We develop the hybrid bacteria foraging algorithm to solve multi objective optimization i.e. minimization of makespan and energy consumption.

- We employ the mutation and crossover technique of genetic algorithm in bacteria foraging algorithm to achieve local and global optimal solution.
- We verify the effectiveness of the proposed multiobjective hybrid bacteria foraging algorithm (MHBFA) specifically its role in solutions' diversity and quality, convergence and stability.

We employ the mutation and crossover technique of genetic algorithm in bacteria foraging algorithm to achieve local and global optimal solution. We verify the effectiveness of the proposed multiobjective hybrid bacteria foraging algorithm (MHBFA) specifically its role in solutions' diversity and quality, convergence and stability. Statistical validation of the obtained results against that of GA, PSO, BFA using significance test.

The rest of the paper is structured as follows. In Section 2, a description of framework of task scheduling algorithm has been presented. System Models and Definition for resource scheduling has been presented in Section 3. Section 4 describes the problem formulation. Section 5 presents the multi-objective approach gives a brief overview of multi-objective approach based on BFA. Section 7 presents the BFA based proposed strategy in detail. Section 8 presents the simulation results. Conclusions and the future works have been presented in Section 9.

2. Framework of task scheduling algorithm

To design and provide scheduling management framework for engineering implementation in IaaS Clouds, in this section, we introduce one of the important aspects in cloud computing resource management, i.e., task scheduling. The goal of cloud computing is to provide an optimal scheduling of the tasks, to provide the users, and the entire cloud system with optimal operation time, improved QoS at the same time and load balancing. Load balancing and task scheduling are closely related with each other in the cloud environment. Task scheduling is for the optimal matching of tasks and resources [14]. To design and provide scheduling management framework for engineering implementation in IaaS Clouds, in this section, we introduce one of the important aspects in cloud computing resource management, i.e., task scheduling. The cloud is mainly to provide users with a Quality of Service (QoS). The main aim of task scheduling algorithms is to achieve two main objectives namely, task scheduling helps to minimize the makespan and energy. Now we briefly depicts the entire energy aware task scheduling framework. This framework is composed of four main components: user portal, information service, task scheduler, and cloud data center with physical machines (PM) Fig. 1. The user portal provides an interface for users to submit task unit. The task unit further divided into small tasks to be executed in PMs.

Information Service keeps the details of resource utilization and other log information to help scheduler to schedule a task to a PM in a data center. The scheduler accepts the task unit from the user portal and uses Information Service to choose the appropriate PM in a cloud data center. After the task unit is complete its execution, the result and the new status of the resource will be sent back to the Information Service for another scheduling.

3. System models and definition

The model of the cloud system to be considered in this work can be described as follows [15]. We describe different models and definitions associated with problems formulated in this paper. In this paper, we have assumed that the cloud scheduling environment is highly heterogeneous and with the Physical machines have uncertain utilization information. We have designed multiple objectives of minimizing energy consumption and makespan. Under the favorable condition we find the Pareto set of multi-objective optimization.

3.1. Definitions

The cloud service provider keeps the detail information about the arrival of user requests and the available utilization of PMs in the data center. The complete scenario can be represented by using a direct acyclic graph where the user requests are presented. Here properties of tasks, task unit relationship and task unit arrived are captured.

The fundamental properties of tasks that we take into consideration are the CPU bounded task that spends most of it's time in computation that is assigned with multiple processing element large RAM size. Whereas the I/O bound tasks dependent only peripheral devices connected to the machines.

Because of which it may need a machine with sufficient network bandwidth and a buffer of large capacity. One important property of the task unit is to add input and output instruction size to reserve the resource available in a PM. There might exist dependencies among the task units. As an example given in Fig. 2 is a DAG, in which each node represents a task unit and its task type, a directed line depicts the dependency relationship between the tasks, and we can add weight to connects two tasks, the edges to represent the flow size. The graph can be represented by using 5-tuples as follows: G = (TD, TS, D, M_i, M_{out}). The semantics and definition of all tuples is given as follows:

Definition 1. User request (TD)**:** This is the set of users request that consists of $1\cdots n$ task units.

Definition 2. Task type (TS)**:** It is the task type of each single task unit from $1\cdots m$, where T_m denotes maximum number of task in a task unit. Form example if we have three task unit $\{TD_1, TD_2, TD_3\}$, then each task unit may have task type $TD_1 = (TS_{1,1}\cdots TS_{1,m})$, $TD_2 = (TS_{2,1}\cdots TS_{2,m})$ and $TD_3 = (TS_{3,1}\cdots TS_{3,m})$.

Definition 3. Task dependency (***D***)**:** It represents the dependencies between the task units in TD. Let $D_{ij} = 1$, that is the data obtained from TD_i is used by TD_j. Otherwise, $D_{ij} = 0$.

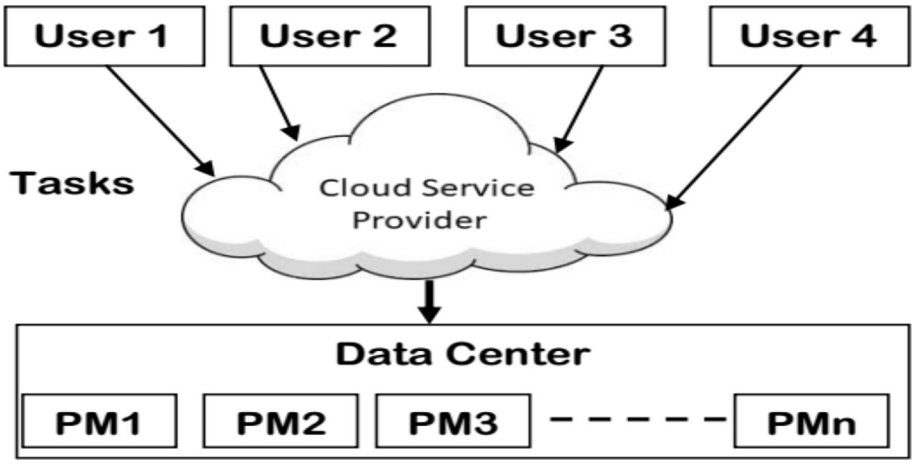

Fig. 1. Cloud architecture.

$$D_{3,3} = \begin{bmatrix} & \mathrm{TD}_1 & \mathrm{TD}_2 & \mathrm{TD}_3 \\ \mathrm{TD}_1 & 0 & 1 & 0 \\ \mathrm{TD}_2 & 1 & 0 & 0 \\ \mathrm{TD}_3 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathrm{ES}_{ij} = \begin{bmatrix} & \mathrm{PM}_1 & \mathrm{PM}_2 & \dots & \mathrm{PM}_j \\ \mathrm{TD}_1 & \mathrm{TS}_{1,1} & \mathrm{TS}_{1,2} & \dots & \mathrm{TS}_{1,j} \\ \mathrm{TD}_2 & TS_{2,1} & \mathrm{TS}_{2,2} & \dots & \mathrm{TS}_{2,j} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{TD}_i & \mathrm{TS}_{i,1} & \mathrm{TS}_{i,2} & \dots & \mathrm{TS}_{i,j} \end{bmatrix}$$

Definition 4. Input data ($M_{\mathbf{in}}$)**:** It represents the input data size of task unit.

Definition 5. Output data ($M_{\mathbf{out}}$)**:** It represents the output data size of task unit. We have assumed that the resource pool is heterogeneous and the resources may be a physical machine or a server or PC in remote that constitute the data center. The same resources have the different configuration. The next outcome is different even when they deal with the same type of task. The overall heterogeneity characteristics can be generalized by varying the network bandwidth and physical machine capacity. The capacity of the PM gives minimum time taken to execute the data present in a task that builds a direct relationship between the CPU capacity and the available memory size.

The network bandwidth facilitates the rate and cost of data communication between two physical machines. But, it does not differentiate the type of tasks rather deal with the only data flow. The resource information, M, consists of six-tuples i.e. M = (PS, CP, R, CE, N_{bw}, E_{com}). The definition of each tuple given as follows.

Definition 6. Physical machines (PS)**:** Let $\mathrm{PS} = (\mathrm{PM}_1, \mathrm{PM}_2, \mathrm{PM}_3, \dots, \mathrm{PM}_n)$ denote the set of physical machines inside a data center.

Definition 7. Computing power (ES)**:** Let ES is a matrix that give the computing power of the PM. The execution time of task unit type i on a physical machine PM_j is denoted as ES_{ij}. The average power of PM_j is denoted as $\mathrm{ES}_{\mathrm{avg},j}$, we calculate $\mathrm{ES}_{\mathrm{avg},j}$ value by calculating the mean of elements in column of matrix ES_j. The detail is explained as follows:

Definition 8. RAM in PM (***R***)**:** It is the available RAM (memory) size of each PM.

Definition 9. Computing energy (CE)**:** CE this is the matrix that gives the rate of consumption of executing a task unit computing energy and, CE_{ij} denotes the energy consumed by a PM_j to execute i task unit type per unit time per unit data.

Definition 10. Bandwidth ($N_{\mathbf{bw}}$)**:** N_{bw} denotes the bandwidth between PMs and, the data transfer rate between PM_i to PM_j is denoted as $N_{\mathrm{bw},ij}$.

Definition 11. Communication energy ($E_{\mathbf{com}}$)**:** E_{com} denotes the energy consumption rate for the communication i.e. the energy consumed during transmission of data from a physical machine PM_i to PM_j per unit time per unit data is denoted as $E_{\mathrm{com},ij}$.

Definition 12. Mapping variable ($\boldsymbol{\chi}$)**:** χ is mapping variable that maps task unit and PMs. $\chi(i) = j$ represents the assignment of task unit i to PM_j to be executed.

3.2. Models

This section highlights the various models and few definitions based upon which the problem is formulated. The two most important objectives we have considered for optimization for cloud computing resource scheduling is the minimization of the makespan and energy consumption. The contradictory nature of these two objectives arises out of parallelism and heterogeneity. The former states that makespan is reduced at the cost of rigorous inter-PM data trans-

mission, which directly affects the total energy consumption in the data center and the later says that the resource that faster in nature is not necessarily the cheapest.

3.2.1. Makespan model

We define makespan as the time taken from the moment a user submits his request to the completion of the last task unit. It includes both the processing time and waiting time. The processing time for the user request is calculated as follows: the user request is decomposed into task unit, and then topological sorting is applied to make sure that every task unit can only be dependent on those with smaller indexes.

The total processing time is virtually the same as with the completion time of task unit TD_i. We calculate the completion time $CT(i)$ for each task unit TD_i, by adding the execution time for the current task unit and the time is taken to arrive all the required data at the current PM.

Consider the DAG depicted in Fig. 2 as an example. We figure out the completion time of task unit TD_8 by adding the completion time of task unit and TD_8 and the processing time of TD_5, TD_6, and TD_7 (i.e., the time when all the input data for task unit TD_8 will arrive). The above information can be mathematically modeled as follows:

$$\mathrm{CT}(i) = T_{\mathrm{aux}} + T_{\mathrm{ex}} \tag{1}$$

where T_{aux} is the time taken to arrive all the required task to the current task, which is given as follows:

$$T_{\mathrm{aux}} = \mathrm{Max}_{j=1}^{i-1}\left\{ D_{ij} \times \mathrm{CT}(j) + \frac{D_{ij} \times M_{\mathrm{out}}}{N_{(\mathrm{BW},p,q)}} \right\}$$

where p and q are defined as follows: $p = \chi(j)$, $q = \chi(i)$ and T_{ex} is the time taken to execute current task, which is given as follows:

$$T_{\mathrm{ex}} = \mathrm{ES}_{(g,h)} M_{i,i} \quad \forall i$$

where g and h are defined as follows: $g = \mathrm{TD}_i$, $h = \chi(i)$.

The values of CT vector's elements may be calculated recursively. When the waiting time ignored, we claim the makespan value of $\mathrm{CT}(n)$ of the user request, where n is the user request's last task unit. The cloud computing works on a concept of virtualization where the hardware abstraction is performed. It works on the principle of creating a virtualized environment, where each physical machine has its running execution environment. Here the scenario creates an illusion in the user's mind that the user owns whole CPU, memory, and other hardware accessories: yet in fact, it is just a virtual environment, and the user still needs to transfer data "from and to" into the physical memory. But when we observe Eq. (1), it holds true only when the PM is directly assigned to a task unit to which it is assigned. The conclusion drawn from the last passage is that the level of multi-threading can not be too high; generally, the CPU would invest a significant measure of time, energy in setting switch and page fault, more regrettable as yet may lead to thrashing.

The waiting time is eventually going to add up with the processing time as the level of multi-threading is not too high when certain PMs are overloaded, or more task units are assigned. A deep analysis of the process reveals that the balancing of load among the PMs present in the data center is an essential attribute for task scheduling. But it is impossible until we have proper information about the distribution of load among the PMs of the data center. Even if this information are measurable, but the cloud broker or resource provider do not make it publicly accessible. To assume more tasks the resource provider tends to understand the process. Because of above such reason we are compelled to find its solution. As we have the prior information on the task yet to be assigned, that is sufficient for predict the load on different PMs even though the information about the already assigned task may not be known to us. It is an effective way but seems to be restrictive. In this proposal, we made an assumption that the ratio of load distribution each PMs average computing power and load distribution [16]. We define the load balance as follows:

$$\mathrm{LB} = \sum_{i=1}^{n} \left(A(i) - B(i)\right)^2 \tag{2}$$

where, n is the number of PMs in the data center

$$A(i) = \frac{\sum_{j=1}^{m} M_{i,j} \big| \chi(j) = i}{\sum_{j=1}^{m} M_{i,j}} \tag{3}$$

$$B(i) = \frac{R_i \big/ \mathrm{EC}_{\mathrm{avg},i}}{\sum_{i=1}^{n} R_i \Big/ \mathrm{EC}_{\mathrm{avg},i}} \tag{4}$$

There is a few risk associated with the deviation from the ideal ratio i.e. some PMs may remain busy compelling other tasks into the waiting queue for a long time that adversely increases the makespan of the system. Therefore we have assumed that the ideal ratio is considered for initial load distribution. Hence the load balancing is a risky parameter which indirectly affects the makespan of the system. So the newly mathematical model for makespan is given as follows:

$$\mathrm{CT_f} = \mathrm{CT}(n) \times e^{\beta} \tag{5}$$

where β is a load balancing factor. β increases with the increase in data traffic and the makespan also increase showing the effect of unequal load distribution. The small value of is the impact of load balance on the makespan is ignorable indicating the idleness of data traffic.

3.2.2. Energy model

The energy consumption in a cloud is the sum of energy consumption individual unit that participates during the servicing of the user requests. The authors in Ref. [17]

proposed an energy model which states that the CPU consumes more energy compared to other equipment or hardware that involve in the task scheduling process. The CPU energy consumption is dependent on the utilization of resources, voltages, and frequencies. Particularly for CPU, energy consumption mostly relies upon its voltage and frequency, which implies, the energy consumption remain unchanged as long as the working state of CPU is constant. Energy consumed during computation and communication can be calculated as follows:

$$E_{\mathrm{C}} = \sum_{i=1}^{n} \mathrm{CE}_{(g,h)} E_{\mathrm{com},(g,h)} M_{i,i} \tag{6}$$

where g and h are defined as follows: $g = \mathrm{TD}_i$, $h = \chi(i)$.

$$E_{\mathrm{CE}} = \sum_{i=1}^{n} \sum_{j=1}^{i-1} \frac{D_{j,i} M_{o,j}}{N_{\mathrm{bw},(p,q)}} \times E_{\mathrm{com},(p,q)} \tag{7}$$

where p and q are defined as follows: $p = \chi(j)$, $q = \chi(i)$ Finally, the mathematical expression for the total energy consumption is the sum of two parts first, communication energy and computation energy which is expressed by the following equation:

$$T_{\mathrm{E}} = E_{\mathrm{C}} + E_{\mathrm{CE}} \tag{8}$$

4. Problem formulation

From the framework of task scheduling algorithm, the following subsection first analyzes the relationship between make span and energy consumption then the objectives formulated.

4.1. A priori analysis

The cloud operation comprises of five stages: the arrival of requests from users, resource exploration that facilitates the resource requirement, scheduling of resource this schedules the resources to execute users requests, service and process monitoring and the last one feedback submission. Out of these steps, the third one plays a vital role in the quality of service and total cost during the user request execution life cycle. The major factor that affects this stage is the makespan and energy consumption. To explore both makespan and energy efficiency, three crucial issues must be addressed. First, excessive power cycling of a server could reduce its reliability. Second, turning resources off in a dynamic environment is risky from the QoS perspective. Due to the variability of the workload and aggressive consolidation, some PMs may not obtain required resources under peak load, and fail to meet the desired QoS. Third, ensuring SLAs brings challenges to accurate application performance management in cloud computing environments. All these issues require effective scheduling policies that can minimize energy consumption and make span without compromising the user-specified QoS requirements. Currently, task scheduling in a Cloud data center aims to provide high performance while meeting SLAs, without focusing on allocating PMs to minimize energy consumption and makespan. Task scheduling in cloud computing is an NP-complete problem [18]. Authors in Ref. [17] propose an energy efficient resource scheduling algorithm that reduces operating costs and provides quality of service. Energy saving and resource utilization are achieved through scheduling of virtual machines. The QoS is modeled by the amount of resource needed for the CPU capacity measured in Millions Instructions Per Second (MIPS), the amount of primary memory (RAM), and by the network bandwidth rate. Insufficient available of these resource leads to SLA violation.

In Ref. [19] authors proposes a load sharing optimization problem between a remote and a local Cloud service. Their multi-objective approach defines to optimize energy consumption per job and response time using a Poisson arrivals jobs rate.

Authors in Refs. [20–22] propose DVFS enabled techniques to minimize power consumption in a distributed system ignoring performance.

From the above study, we observe that it is a trade-off to minimize makespan and energy. So, minimization of both these conflicting parameters at topological sorting to minimize both makespan and energy can be better realized as the multi-objective optimization problem which is discussed below.

4.2. Problem objective

In this section, the overall problem is defined and named as Multi objective optimization of task scheduling algorithm (MOOTS) problem.

4.3. Objective functions

Based on Definition and Eqs. (1)–(8) given in previous section, our objectives are defined as follows.
Objective (1): Minimization of makespan $(\mathrm{CT_f})$

$$\mathrm{Min}\left(\mathrm{CT}(n) \times e^{\beta \times LB}\right) \tag{9}$$

Objective (2): Minimization of energy (T_{E})

$$\mathrm{Min}(T_{\mathrm{E}}) \tag{10}$$

$$\text{Fitness function} \quad \mu = \alpha\left(\mathrm{CT}(n) \times e^{\beta \times LB}\right) + \beta(T_{\mathrm{E}}) \tag{11}$$

where $0 \le \alpha < 1$ and $0 \le \beta < 1$ are weights to prioritize components of the fitness function.

4.4. Constraints

The following constraints are considered.

Constraint 1. This constraint confirms that each task unit can only select one physical machine from the resource pool available in cloud data center. This is given as follows.

$$\chi(i) \in \{1,2,\ldots,n\} \ \forall i \ \{1,2,\ldots,m\} \tag{12}$$

where n is task units and m is number of task per task unit. The maximum time required to process a user's task unit.

Constraint 2. Each task unit's maximum processing time;

$$\mathrm{CP}_{\mathrm{TS}_i\chi(i)} M_{i,i} \le T^{\mathrm{p}}_{\max} \ \forall(i) \tag{13}$$

where $T^{\mathrm{p}}_{\max}$ is the maximum processing time.

Constraint 3. Each task unit maximum communication time;

$$\mathrm{MAX}_{j=0}^{i-1}\left\{\frac{D_{ij}\times M_{o,j}}{N_{\mathrm{bw},\chi(j)\chi(i)}}\right\} \le T^{\mathrm{c}}_{\max} \ \forall(j) \tag{14}$$

where j is task unit, i is the number of PMs and, $T^{\mathrm{c}}_{\max}$ is the maximum communication time.

Constraint 4. Each task unit maximum processing energy consumption;

$$\mathrm{CE}_{\mathrm{TS}_i\chi(i)}\mathrm{CP}_{\mathrm{TS}_i\chi(i)} M_{i,i} \le E^{\mathrm{p}}_{\max} \ \forall(i) \tag{15}$$

where $E^{\mathrm{p}}_{\max}$ is the processing energy.

Constraint 5. Energy consumption during communication by each task unit.

$$\sum_{i=1}^{j} \frac{D_{ij}M_{o,i}}{N_{\mathrm{bw},\chi(i)\chi(j)/E_{\mathrm{com}}\chi(i)\chi(j)}} \le E^{\mathrm{c}}_{\max} \ \forall(j) \tag{16}$$

where $E^{\mathrm{c}}_{\max}$ is the communication energy.

Constraint 6. For the proper utilization of PMs the load distribution should be some threshold value for it, and in our case, the lower threshold limit is 0, and the upper limit is as given in the following equation;

$$\frac{\sum_{i=1}^{n} M_{i,i}\big|\chi(i)=k}{\sum_{i=1}^{n} M_{i,i}} \le \mathrm{Th}^{\mathrm{u}} \ \forall(k) \tag{17}$$

where Th^{u} is the upper threshold for executing task.

4.5. Intractability of MHBFA problem

Definition 1. Let $L=(x_1,\ldots,x_{ji},\ldots,x_n)$ be a given list of n items with a value of $x_{ji}\in(0,1]$, and $B=b_1,\ldots,b_m$ be a finite sequence of m bins each of unit capacity. The bin-packing problem is to assign each x_{ji} into a unique bin, with the sum of numbers in each $b_j\in B$ not exceeding one, such that the total number of used bins is a minimum (denoted by L^*) [23,24].

Proposition 1. The optimization problem described in Eq. (11) is an NP-hard problem.

Proof. This proposition can be easily proven by reducing the problem to the bin-packing problem [23,24], which is a well-known NP-hard problem. The number of bins m is equal to the available N data centers. The dimensions of an application j consist of two parameters d_j and e_{ji}. However, e_{ji} depends on the frequency of the CPUs of data center i. By defining a transformation function $\rho: R\times R\rightarrow R$, we can transform e_{ji} to e_j. This restriction only considers data centers with the same frequency for all CPUs. Consequently, $f(d_j,e_j)=x_{ji}$ and by Definition 1, it is a bin-packing problem. □

5. Multiobjective approach

5.1. Preliminaries and background

In general, the multi-objective optimization problem involves multiple conflicting objectives. To obtain the solution for multi objective challenges the objectives are aggregated to scalar function and these single objective optimization problems are solved. In this paper we find out Pareto optimal set which is an optimal trade off. The detail scenario and local and global Pareto optimal set difference have been defined below.

In an multiobjective optimisation problem partially order conveying that two solutions are related to each other. One way is that one objective dominates over other and second case no one dominates.

Definition 1. Let us consider a case of multiobjective minimization problem. In this scenario we have m decision variable and n objectives.

$$\text{Minimize } y=\{f_1(x),f_2(x),\ldots,f_n(x)\} \tag{18}$$

where $x=x_1,x_2,\ldots x_m\in x$ is called decision vector, x is parameter space, y is objective vector. A decision vector $a\in x$ is said to dominate a decision vector $b\in x(S)$ if and only if

$$\forall i\in\{1,\ldots,n\}\ f_i(x)\le f_i(x)$$

$$\exists i\in\{1,\ldots,n\}\ f_i(x)\le f_i(x) \tag{19}$$

Based on the above relation, we can define non-dominated and Pareto-optimal solutions.

Definition 2. Let $a\in x$ be an arbitrary decision vector. Pareto optimal decision vectors cannot be improved in any objective without causing degradation in at least one other objective. In our terminology, they represent, globally optimal solutions. However, analogous to single objective optimization problems, there may also be local optima which constitute a non-dominated set within a certain neighborhood.

This corresponds to the concepts of global and local Pareto-optimal sets introduced by Deb [25].Note that a global Pareto-optimal set does not necessarily contain all Pareto-optimal solutions. If refer to the entirety of the Pareto-optimal solutions, simply write "Pareto-optimal set". The corresponding set of objective vectors is denoted as "Pareto optimal front". To solve multi-objective optimization problem by BFA, some variants of BFA and hybrid BFA have been proposed. Authors in Refs. [26–28] have proposed Multi-objective Bacterial Foraging Optimization (MBFA). The draw back of their proposal is that it stuck in local optima as their basic technique is derived from BFA. However many of the approach now a days ignores multi-dimensional approach of the problem. Authors in Refs. [28–31] have proposed hybrid HBFA task scheduling algorithm considering one dimensional approach. In the proposed work we include multi-objective optimization with integration between health sorting approach and Pareto dominance mechanism, where the main goal of multi-objective optimization problems is to obtain a non-dominated front which is close to the true Pareto front.

6. Principle of HBFA

The genetic algorithm (GA) has lack of local search capability and excellent global search. The ability of Bacteria Foraging (BF) global search is poor and very high local search capability [32–38]. When we combine these two algorithms through the selective combination of certain favorable function, this could give rise a solution that may contain best local and global search capability and faster convergence time. The HBF posses all the merged properties of GA and BF [30,39,40]. The GBF appears to be an implementation of BF than GA, but it combines the features of both the algorithm. Literature in reviewed that BF is hybridized with other algorithms other than GA and theoretically verified the effectiveness of the developed algorithm. In all these literature it has been observed that the HBF has maintained general validity and optimized features can be applied in many other applications.

The HBF inherits both swarming, and elimination and dispersal from BF. The objective is to make BF more global concerning search capability so we need to keep these features with HBF and change those function which does not support global search capability. The elimination and dispersal and swarming are the procedure which is critical in globalization search procedure hence they have been kept in the procedure. The other two functions, i.e., chemotaxis and reproduction are in focus to convert into global searching capability.

The concept of genetic algorithm was introduced by Holland in 1973, which is inspired by biological evolution. It is a random search algorithm achieved by simulating natural selection and genetic mechanisms. Genetic algorithm simulates the basic process of biological evolution with a string of digital to analog the individual chromosome organisms, and the basic processes of biological evolution through selection, crossover and mutation operators. The fitness function represents the quality of the solutions, the average fitness of each generation can be increased through the continuous upgrading of the population, the direction of the evolution of the population can be guided through fitness function and on this basis we can make the solution represented by the optimal individual approximate the global optimal solution.

7. Proposed MHBFA for scheduling problem

This section provides the proposed MHBFA bacteria foraging optimization algorithm to solve task scheduling in cloud computing. The evaluation of the objective function is proceeded in each iterative steps which lead to obtain better solution to the multi objective optimization problem. The position (coordinate) of the bacteria represents the parameters to be optimized. In a simple sentence, we can say that a bacteria represents the solution for our task scheduling problem. We have generated many bacteria for the algorithm input. The bacteria are evaluated against the objective function to obtain minimum makespan and energy.

The parameters are discretized in the desirable range, each set of these discrete values represent a point in the space coordinates. These parameters are discretized in the desirable range. Each of these discrete values describes a point on a space coordinate. In the proposed MHBFA algorithm, at each iteration, all bacteria are evaluated according to a measure of solution quality.

Our primary objective is to reduce makespan and energy consumption, which is bacteria position. The parameters used in algorithm's are as follows: p: Dimension of the search

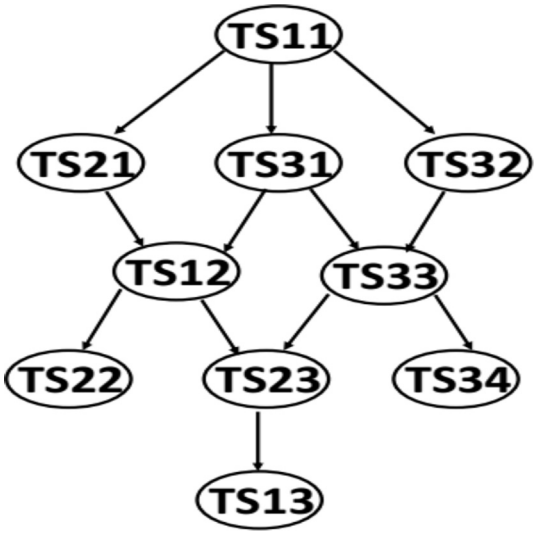

Fig. 2. DAG of task unit and tasks.

Table 1
Task units and their tasks.

	TSi1	TSi2	TSi3	TSi4
TD1	TS11	TS12	TS13	
TD2	TS21	TS22	TS23	
TD3	TS31	TS32	TS33	TS34

space; S: the number of bacteria in the population; $C(i)$: it is random direction taken during tumble; N_c : chemotaxis steps; N_s : swimming length; N_{re} : reproduction steps; N_{ed} : elimination and dispersal events; P_{ed} : elimination and dispersal probability.

7.1. Initial position generation

The task unit details derived from the DAG of Fig. 2 is given as follows (see Table 1).

The bacteria holds the solution of the problem domain which is the novelty associated with the proposed algorithm. For the proposed problem, we obtain the intermediate representation of solution as follows: The bacterium is encoded with two variable η and λ which denotes the linear vector and the other one is a matrix. The η give the scheduling sequence of all tasks on to the PMs. The λ indicates the mapping of tasks to PMs.

7.2. Bacterial representation

The representation of bacteria which is a potential solution of the identified problem related to energy consumption is shown in Fig. 3.

The set of bacteria consists of $S = S_1, S_2, \ldots, S_n$. Each bacteria is encoded with η and λ as follows.

Given tasks as given in Definition 3.1, the η can be denoted as follows.

$$\eta_n = \{\mathrm{TD}_{n1}, \mathrm{TD}_{n2}, \ldots, \mathrm{TD}_{nk}\} \tag{20}$$

The η follows restriction of DAG and also it is the permutation of task numbers. The initial contents of the η are generated using a breadth-first search procedure over the DAG. The η contains task numbers. The repetition of the task number in the η is not allowed. The importance of the η is that it specifies the order of execution of tasks in the cloud resource. Suppose we are given η, resources $\mathrm{PS} = \{\mathrm{PM}_1, \mathrm{PM}_2, \mathrm{PM}_3, \ldots, \mathrm{PM}_m\}$, the λ can be expressed as follows.

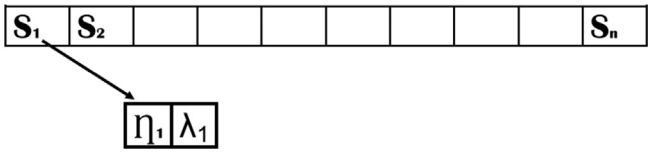

Fig. 3. Structure of a bacteria.

$$\lambda = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2m} \\ & & \ldots & & \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{nm} \end{bmatrix}$$

The possibility of assignment of task TS_i to the PM. PM_j is denoted as x_{ij}. We assign the value to λ from the uniform distribution on the interval [0,1], satisfying the following condition:

$$x_{i,j} \in [0,1],\ i \in \{1,2,\ldots,n\},\ j \in \{1,2,\ldots,m\} \tag{21}$$

$$\sum_{j=1}^{m} x_{i,j} = 1,\ i \in \{1,2,\ldots,n\},\ j \in \{1,2,\ldots,m\} \tag{22}$$

The allocation matrix value remains between 0 and 1. When the value exceeds this range, then we set it to the closest boundary value. The formula given in (21) facilitates this. When we observe formula (22), it gives a sum of all probability values of allocation TD_i, where i is the task, it also affirms that the value should not be greater than 1. While updating the λ value formula (22) could be violated, so we normalize the value by using formula (23).

$$x_{i,j} = \frac{x_{i,j}}{\sum_{k=1}^{n}} \tag{23}$$

To select a resource for scheduling a task TD_i, we check the highest probable value in the row to find the match.

$$n_i = \max\left(x_{i,j}\right),\ i \in \{1,2,\ldots,n\} \tag{24}$$

The example of λ with four PMs and tasks given in the DAG is given as follows:

$$\lambda = \begin{array}{c} \\ TS11 \\ TS12 \\ TS13 \\ TS21 \\ TS22 \\ TS23 \\ TS31 \\ TS32 \\ TS33 \\ TS34 \end{array} \begin{array}{c} \begin{array}{cccc} PM1 & PM2 & PM3 & PM4 \end{array} \\ \begin{pmatrix} 0.12 & 0.26 & 0.46 & 0.16 \\ 0.23 & 0.19 & 0.34 & 0.24 \\ 0.32 & 0.06 & 0.30 & 0.33 \\ 0.26 & 0.28 & 0.25 & 0.21 \\ 0.44 & 0.12 & 0.43 & 0.01 \\ 0.38 & 0.31 & 0.05 & 0.26 \\ 0.27 & 0.33 & 0.25 & 0.16 \\ 0.62 & 0.16 & 0.12 & 0.09 \\ 0.31 & 0.20 & 0.11 & 0.37 \\ 0.13 & 0.10 & 0.54 & 0.23 \end{pmatrix} \end{array}$$

Based on the λ value the χ (mapping of tasks to PMs) is given in Table 2.

7.3. Hybrid chemotaxis

Hybrid chemotaxis is a process where the *E. coli* bacteria is carried in flagella. It comprises of three basic steps i.e.

Table 2
χ value.

PM3	PM3	PM4	PM2	PM1	PM1	PM2	PM1	PM4	PM3
TS11	TS12	TS13	TS21	TS22	TS23	TS31	TS32	TS33	TS34

tumbling, swimming and mutation. By tumbling (movement in the random direction) and swimming (motion in the same direction) the bacteria moves ahead to collect nutrients. We add the mutation step of genetic algorithm to make the process of chemotaxis as hybrid chemotaxis.

Here we initialize the bacteria set S_n. We initiate a loop to find the fitness function for each bacteria in the bacteria set. Extract the first bacteria from the set i.e.η_1 and λ_1, find the fitness function μ as $J(i,j,k,l)$ that corresponds initial position vector $\theta(i,j,k,l)$ in jth hybrid chemotaxis, kth reproduction and lth elimination and dispersal step. In this paper we have considered $\theta(i,j,k,l)$ as the current λ and η value. Assign $J_{\text{last}}(i,j,k,l) = J(i,j,k,l)$. Next, calculate $\theta(i,j+1,k,l)$ for $(j+1)$th hybrid chemotaxis step as follows:

$$\theta(i,j+1,k,l) = \theta(i,j,k,l) + C(i)\phi_j \tag{25}$$

$\theta(i,j+1,k,l)$ gives rise the fitness function at $(j+1)$th hybrid chemotaxis step as $J(i,j+1,k,l)$. $C(i)$ is the size of the step taken in the random direction specified by the tumble. Generate a random direction ϕ_j, which is a value between interval $[-1,1]$. It is the direction angle of the jth hybrid chemotaxis step.

If the $J(i,j+1,k,l)$ at $\theta(i,j+1,k,l)$ is better than the $J(i,j,k,l)$ at $\theta(i,j,k,l)$ then the bacterium takes another step of size $C(i)$, otherwise it tumbles in a random direction of size ϕ_j. This process continue until $j = N_c$. During the chemotactic process the bacterium cell alternate between swims and tumbles. The steps involved in hybrid chemotaxis are presented in Algorithm 4. The swim, move and tumble process carried out as follows:

7.3.1. Tumble

In this process, the bacteria tumble in random direction. We calculate $J(i,j,k,l)$ at new position and assign this value to $J_{\text{last}}(i,j,k,l)$. The new position is calculated based on Eq. (25).

7.3.2. Move

In this step we update the position as $\lambda(i,j+1,k,l)$ to get new values of λ as follows: Let $\lambda(i,j+1,k,l) = \lambda(i,j,k,l) + C(i)\phi(j)$, as this results in a step of size $C(i)$ in the direction of the tumble for bacterium j. Normalize λ according to formula (23). Calculate fitness function $J(i,j+1,k,l)$.

7.3.3. Swim

The swimming and tumbling of the bacteria depends on the value of $J(i,j+1,k,l)$ and $J(i,j,k,l)$, i.e. If $J(i,j+1,k,l)$ greater than $J(i,j,k,l)$, the bacteria swims in step size of $C(i)$ else tumble in random direction of ϕ_j steps.

Update the $\theta(i,j+1,k,l)$. Here we only update λ. The significance of this updation process is that we get different χ value but the η value remains same. The updation process is as follows, i.e. λ is updated through mutation as given in following example.

Table 3
χ value before mutation.

TS11	TS21	TS32	TS31	TS33	TS34	TS12	TS23	TS22	TS13
PM3	PM3	PM4	PM2	PM1	PM1	PM2	PM1	PM4	PM3

Mutation (updation of λ): Mutation to be done in λ to generate new λ'.

$$\text{Before mutation } \lambda = \begin{bmatrix} 0.12 & 0.26 & 0.46 & 0.16 \\ 0.23 & 0.19 & 0.34 & 0.24 \\ 0.32 & 0.06 & 0.30 & 0.33 \\ 0.26 & 0.28 & 0.25 & 0.21 \\ 0.44 & 0.12 & 0.43 & 0.01 \\ 0.38 & 0.31 & 0.05 & 0.26 \\ 0.27 & 0.33 & 0.25 & 0.16 \\ 0.62 & 0.16 & 0.12 & 0.09 \\ 0.31 & 0.20 & 0.11 & 0.37 \\ 0.13 & 0.10 & 0.54 & 0.23 \end{bmatrix}; \text{ and}$$

$$\text{after mutation } \lambda' = \begin{bmatrix} 0.46 & 0.26 & 0.12 & 0.16 \\ 0.23 & 0.34 & 0.19 & 0.24 \\ 0.32 & 0.06 & 0.30 & 0.33 \\ 0.26 & 0.28 & 0.25 & 0.21 \\ 0.44 & 0.12 & 0.43 & 0.01 \\ 0.38 & 0.31 & 0.05 & 0.26 \\ 0.27 & 0.33 & 0.25 & 0.16 \\ 0.09 & 0.16 & 0.12 & 0.62 \\ 0.31 & 0.20 & 0.11 & 0.37 \\ 0.13 & 0.10 & 0.54 & 0.23 \end{bmatrix}$$

Table 3 gives χ value i.e. task allocation to PM before mutation, that corresponds to λ value.

Table 4 gives χ value i.e. task allocation to PM after mutation, that corresponds to λ value.

The process continues till maximum swimming length reaches to N_s. If at any point $(J(i,j+1,k,l) > J_{(\text{last})}(i,j,k,l))$ then the swimming process stops and the chemotactic process continue for next bacterium.

An individual chemotactic step concludes with the movement of a bacteria to a new coordinate position, also known as the space points. The point in the space defines the set of a set of μ value. For each of the points, the fitness function is evaluated to decide the net movement in the solution space. We represent the fitness of ith bacterium's cost function $J(i,j,k,l)$. When we complete chemotactic process then we move to reproduction process as described in next section.

Table 4
χ value after mutation.

TS11	TS21	TS32	TS31	TS33	TS34	TS12	TS23	TS22	TS13
PM1	PM2	PM4	PM2	PM1	PM1	PM2	PM4	PM4	PM3

Algorithm 1: Hybrid chemotaxis algorithm

Input : All bacteria, $J_{(last)}(i,j,k,l)$.
Output: Accumulation of nutrients for each bacteria

```
begin
  while (i < N_b) do
    /* Hybrid chemotaxis starts                                   */
    Tumble: Generate a random direction φ_(i), (random value between [-1,1]);
    {
    Move: Let θ^i(j+1,k,l) = θ^i(j,k,l) + C(i)φ_(i);
    Calculate objective function J(i,j,k,l);
    if (J(i,j,k,l) ≤ J_(last)(i,j,k,l)) then
      J_(last)(i,j,k,l) = J(i,j,k,l);
      J_(health)(i,j,k,l) = J_(health)(i,j,k,l) + J_last(i,j,k,l);
      Swim: let m = 0 /* counter for swim length                  */
      θ^i(j+1,k,l) = θ^i(j,k,l) + C(i)φ_(i);
      while (m < N_s) do
        m = m + 1;
        Calculate objective function J(i,j,k,l);
        J_(swim)(i,j,k,l) = J(i,j,k,l);
        if (J_(swim)(i,j+1,k,l) ≤ J_(last)(i,j,k,l) & m < N_s) then
          J_(last)(i,j,k,l) = J_(swim)(i,j+1,k,l);
          J_(health)(i,j,k,l) = J_(health)(i,j,k,l) + J_(last)(i,j,k,l);
          θ^i(j+1,k,l) = θ^i(j,k,l) + C(i)φ_(i);
          m=m+1;
        else
          if (J_(swim)(i,j+1,k,l) ≥ J_(last)(i,j,k,l) then
            Mutate: Mutation to be done in θ^i(j+1,k,l);
          if m == N_s then
            i=i+1;
    else
      if j < N_c then
        j=j+1;
      if j == N_c then
        i=i+1;
    }
end
```

7.4. Hybrid-reproduction

The reproduction process continues in three steps viz. selection, crossover and mutation. In this paper $J^i_{(\text{health})}$ denotes the accumulated fitness function value during the bacterium lifetime of bacteria i. Sort bacteria in order of ascending cost $J^i_{(\text{health})}$.

7.4.1. Selection

Select the bacterial population and sort them in increasing order on the basis of minimum J_{health} (which is the health of the ith bacteria). The J_{health} is calculated as follows: $J^i_{(\text{health})} = \sum_{j=1}^{N_c+1} J(i,j,k,l)$. Each $J^i_{(\text{health})}$ is the fitness value (μ) of bacteria i.

7.4.2. Crossover

The cross over is performed with the η value. The significance of this crossover process is that it gives new task execution order. It is done randomly in the bacterial population with the least fit in nature. The updation of η is done as given in Ref. [41] i.e. randomly choose as explained in following example η from bacterium population. A new η is generated through crossover operation, as follows. The crossover operation randomly chooses other bacterium's η to form a pair of η. It randomly marks a cut-off point for each pair, which divides the η of the pair into two parts. Then the tasks in each second part are reordered. Fig. 4 shows a η crossover process where the new ordering of the tasks in one's second part to the relative position of these tasks in the other original scheduling vector in the pair.

Algorithm 2: Hybrid chemotaxis algorithm

Input : All bacteria, $J_{(health)}$.
Output: Bacteria

```
begin
  Select: Sort the bacteria on the basis of J_health accumulated during chemotaxis step;
  Crossover: Perform crossover with leastfit bacteria in the colony;
  Mutate: Perform mutation in the position of the bacteria ;
end
```

Because η is discrete value, the swarm effects does not consider the scheduling vector.

7.4.3. Mutation

The mutation is performed by taking the λ value of least cost J_{health} value as done in previous chemotactic steps. The outcome of the reproduction step gives rise bacterias, i.e., half of the population of total bacteria. The discarded bacteria have higher $J_{(\text{health})}$ values compared to the one selected for future production. The newly selected bacterias further split to fill the vacant places of discarded bacteria in bacteria population. This process again moves to chemotactic step as described above. The process continues until $k < N_{\text{re}}$.

7.5. Elimination-dispersal

The bacterias dispersed to the random location in the optimization after elimination. The elimination is done with the probability P_{ed} to keep the bacteria population constant. This process again moves to reproduction step as described above. The process continues until $k < N_{\text{ed}}$.

Algorithm 3: MHBFA based task scheduling algorithm

Input : Input Bacteria
Output: Mapping of each tasks to resources

begin
  Initialize parameters P, N, N_c, N_s, N_{re}, N_{ed}, P_{ed}, $C(i)$ (i = 1,2, . . . , S) ;
  /* Elimination and dispersal loop starts */
  while $(l + 1 < N_{ed})$ **do**
    /* Reproduction loop starts */
    while $(k + 1 < N_{re})$ **do**
      /* Hybrid chemotaxis loop starts */
      while $(k + 1 < N_c)$ **do**
        for $i \leftarrow 1$ **to** S **do**
          ; /* Hybrid chemotaxis starts */
          Tumble: Generate a random direction $\phi_(i)$, (random value between [-1,1]);
          {
          Move: Let $\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi_(i)$;
          Calculate objective function $J(i,j,k,l)$;
          if $(J(i,j,k,l) \leq J_{(last)}(i,j,k,l))$ **then**
            $J_{(last)}(i,j,k,l) = J(i,j,k,l)$;
            $J_{(health)}(i,j,k,l) = J_{(health)}(i,j,k,l) + J_{last}(i,j,k,l)$;
            Swim: let $m = 0$ /* counter for swim length */
            $\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi_(i)$;
            while $(m < N_s)$ **do**
              $m = m + 1$;
              Calculate objective function $J(i,j,k,l)$;
              $J_{(swim)}(i,j,k,l) = J(i,j,k,l)$;
              if $(J_{(swim)}(i,j+1,k,l) \leq J_{(last)}(i,j,k,l)\,\&\, m < N_s)$ **then**
                $J_{(last)}(i,j,k,l) = J_{(swim)}(i,j+1,k,l)$;
                $J_{(health)}(i,j,k,l) = J_{(health)}(i,j,k,l) + J_{(last)}(i,j,k,l)$;
                $\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi_(i)$;
                m=m+1;
              else
                if $(J_{(swim)}(i,j+1,k,l) \geq J_{(last)}(i,j,k,l)$ **then**
                  Mutate: Mutation to be done in $\theta^i(j+1,k,l)$;
                if $m == N_s$ **then**
                  i=i+1;
          else
            if $j < N_c$ **then**
              j=j+1;
            if $j == N_c$ **then**
              i=i+1;
          }
      ; /* Hybrid reproduction starts */
       Reproduction:;
      Select: Sort the bacteria on the basis of J_{health} accumulated during chemotaxis step;
      Crossover: Perform crossover with leastfit bacteria in the colony;
      Mutate: Perform mutation in the position of the bacteria ;
    Elimination-dispersal:;
    with probability P_{ed}, eliminate and disperse each bacterium;
  Terminate:;
  End the Program and output the best performing bacteria position.
end

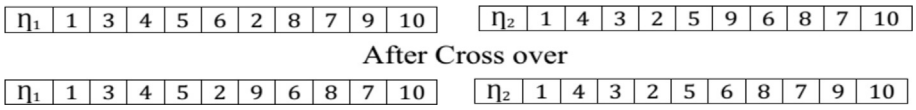

Fig. 4. Crossover operation.

7.6. Termination

In this step the algorithm output $\theta(i,j,k,l)$ and η and terminate the process. The pseudo code of our bacterial foraging algorithm is given in Algorithm 3.

8. Simulation results

In this section, we present the simulation result and discuss the performance of the proposed MHBFA solving MOO problem to verify its effectiveness by comparing the results with other existing heuristic algorithms namely PSO [42], GA [15], BFA [27]. We have presented the simulation result using [15,43,44] simulation with the Matlab R2013a software platform. Our simulation process is of two fold. In the first phase we present experimental results and in second phase we make statistical result comparison. The parameter settings to conduct simulation for the algorithms are given in Table 5.

We have considered three examples and simulated these using our simulation environment. This example consists of 5, 10 and 15 task unit with each task unit different tasks number and five physical resources (PM) for task scheduling and resource allocation on cloud computing. The detail parameter and values and the characteristics of PMs and task unit, that we have used for all our experiments is depicted in Table 6.

We have the five-task unit with 23 tasks, ten task unit with 100 tasks and fifteen task unit with 200 tasks. The first example with five task units are as follow: TD_1, TD_2, TD_3, TD_4 and TD_5, where $TD_1 = (TS_{11}, TS_{12}, TS_{13}, TS_{14}, TS_{15})$ and the resource set are as $(PM_1, PM_2, PM_3, PM_4, PM_5)$. The details are given in Table 7. M_{in} and M_{out} matrix σ (in terabytes) of task unit is given as follows.

Let N_{bw}, the transfer rate be 100 Mbps for uploading and downloading. The receiving time for a task is obtained by dividing M_{in} and M_{out} matrix with N_{bw}. The mapping variable χ and matrix D is obtained while running the process and not mentioned due to space problem. The details of ten and fifteen task unit is also follows same as five task unit. Our simulation process is of two fold. In the first phase we present experimental results and in second phase we make statistical result comparison of the results.

8.1. Experimental results

In this experiment we have consider heterogeneous resources. As the targeted system is a generic Cloud computing environment, i.e. Infrastructure-as-a-Service (IaaS), it is essential to evaluate it on a large-scale virtualized data center infrastructure. However, it is extremely difficult to conduct repeatable large-scale experiments on a real infrastructure, which is required to evaluate and compare the proposed resource management algorithms. Therefore, to ensure the repeatability of experiments, simulations have been chosen as a way to evaluate the performance of the proposed heuristics. To measure the efficiency of the proposed MHBFA algorithm, we configure the performance evaluation criteria based on the parameter defined in the given Table 5. The performance index we considered is makespan that indicates execution time and energy consumption, which is energy consumed by a PM. Table 6 gives the detail of evaluation. The unit of makespan and energy are in minutes and Joule.

We set the environment as low PM heterogeneity with nonuniform and uniform parameters, to calculate makespan and energy consumption of the Cloud task units. The efficiency of an algorithm is measured, by analyzing how it responds to the different heterogeneity of tasks and resources. A comparison of both energy consumption and makespan for

$$\sigma = \begin{array}{c} \\ TSi1 \\ TSi2 \\ TSi3 \\ TSi4 \\ TSi5 \end{array} \begin{array}{c} \begin{array}{ccccc} TD1 & TD2 & TD3 & TD4 & TD5 \end{array} \\ \begin{pmatrix} 0.697 & 0.737 & 0.609 & 0.535 & 0.553 \\ 0.317 & 0.277 & 0.402 & 0.124 & 0.641 \\ 0.983 & 0.654 & 0.315 & 0.057 & 0.336 \\ 0.142 & 0.492 & 0.754 & 0.565 & 0.667 \\ 0.263 & 0 & 0.564 & 0.974 & 0 \end{pmatrix} \end{array} \qquad ES_{i,j} = \begin{bmatrix} 1.239 & 0.836 & 1.195 & 2.645 & 1.304 \\ 0.564 & 0.38 & 0.544 & 1.203 & 0.593 \\ 1.749 & 1.18 & 1.687 & 3.734 & 1.84 \\ 0.252 & 0.17 & 0.243 & 0.539 & 0.265 \\ 0.469 & 0.316 & 0.452 & 1 & 0.493 \end{bmatrix}$$

From the above available data, we compute the ratio of workload and computing capacity of PM to generate $ES_{i,j}$ matrix. The rows and columns of $ES_{i,j}$ matrix gives the value of execution time of a task in each PM and time taken by a PM to execute the task. The $ES_{i,j}$ matrix detail is given as follows. Where the column consists of all five PMs and row consists of TD1 only.

low PM heterogeneity has been shown for uniform and nonuniform parameters.

Fig. 6 shows the makespan comparison for nonuniform and uniform parameters with low machine heterogeneity. From the figure, we observe that the makespan is least in the case of MHBFA for both the non-uniform and uniform parameters.

Table 5
Values considered for different parameter.

Algorithm	Parameter name	Parameter value
PSO	Swarm size	50
	Self-recognition coefficient	2
	Social coefficient	2
	Inertial weight	0.9
BFA	Population size	50
	Elimination-dispersal steps	2
	Reproduction steps	4
	Chemotaxis steps	70
	Maximum swim steps	4
	Step size	0.1
	Elimination-dispersal probability	0.25
	Attraction depth	0.1
	Attraction width	0.2
	Repellant depth	0.1
	Repellant width	10
GA	Number of cloudlets	10–30
	Number of processors	5
	Number of iterations	30
	Population size	10
	Crossover type	Two-point crossover
	Crossover probability	0.5
	Mutation type	Simple swap
	Mutation probability	0.6
	Termination condition	Number of iterations

This is so, as the low variation in execution time across PMs and we have obtained the filtered PM list through PM provisioning.

We have simulated the low PM heterogeneity by taking five, ten and fifteen PMs. Fig. 5 shows the effect on energy consumption by the four heuristics in case of low PM heterogeneity with nonuniform and uniform parameters. From Fig. 5 we observe that MHBFA achieves minimum energy consumption as compared to GA, PSO and BFA resulted in the highest energy consumption in all cases of 5, 10 and 15 PMs. While in case of low PM heterogeneity with the uniform matrix similar conditions are found. When we look the case of low PM heterogeneity, MHBFA performs better than other algorithms viz. GA, PSO, and BFA for both uniform and nonuniform parameters. The other three algorithm GA, PSO and BFA have higher energy consumption compared to MHBFA.

This likewise exhibits the viability of the MHBFA in dealing with the time prerequisite of the user. As said before, tasks which are sent to the Cloud are expected to be autonomous of each other. The attributes of tasks sent to the Cloud to think about the makespan of various algorithms. The outcomes demonstrate that

Table 6
Scheduling parameters and their values.

Parameter	Value
Number of PMs	5–15
Number task unit	5–20
Number tasks per task unit	20–200
Size of task	1000–6000 TB
Bandwidth	100 Mbps
PM ratings	0.10–0.30 Mbps
Energy consumption	0.1–0.3 J/min

Table 7
Task unit and its tasks details.

TD1	TD2	TD3	TD4	TD5
TS11	TS21	TS31	TS41	TS51
TS12	TS22	TS32	TS42	TS52
TS13	TS23	TS33	TS43	TS53
TS14	TS24	TS34	TS44	TS54
TS15		TS35	TS45	

on account of GA, PSO, and BFA algorithms when we send a similar number of tasks/task unit to the Cloud, makespan, and energy increase whereas in the case of the MHBFA makespan and energy utilization decreases in all instances of 5,10 and 15 PMs.

From Fig. 5 we observe the energy consumption comparison in low machine heterogeneity environment for nonuniform uniform parameters. The figure depicts that the energy consumption is low in MHBFA for both the uniform and nonuniform parameters as compared to GA, PSO and BFA. The reason behind this is the PM list that is obtained from the PM provisioning unit is already filtered that gives rise low variation in execution time across various PMs present in the data center. One more important point to be observed here is that the requirement of time management by MHBFA to minimize energy consumption. The results also depicts that in the case of GA, PSO and BFA algorithms, when we send similar task unit to the cloud data center, energy consumption and makespan increases where the same parameter decreases in case of MHBFA.

Here, we have assessed the makespan and energy consumption of the Cloud task units for large PM heterogeneity case with nonuniform and uniform frameworks. We simulate, high PM heterogeneity having an arbitrary number of PMs between 5, 10 and 15. Fig. 8 demonstrates the makespan for nonuniform and uniform cases with high machine heterogeneity. We observe from Fig. 8 that, the execution of the MABFA heuristic is far superior to that of GA, PSO and BFA for the instance of nonuniform and high machine heterogeneity.

Fig. 7 demonstrates the energy consumption for nonuniform and uniform lattices with high machine heterogeneity. The energy consumption decrease in MHBFA and minimum in contrast with GA, BFA, and PSO. Here, we observe that when a similar number of tasks units sent, the energy consumption and makespan are the minimal for the proposed calculation. This is because more task units allocated on less expensive and efficient PM, because of the capacity of MABFA to discover optimal PM. By analyzing the results in Figs. 5–8, we conclude that the MHBFA heuristic outperforms all the other approaches in the cases with both low or high machine heterogeneity.

8.2. Statistical analysis of results

There are many performance indices available to measure the quality of multi-objective optimization algorithms [25,45–49]. The following four indices are often used viz. the converge ratio of two sets (C), the distance-based distribution (D), the maximum spread (SP), and hyper area (HA) (or

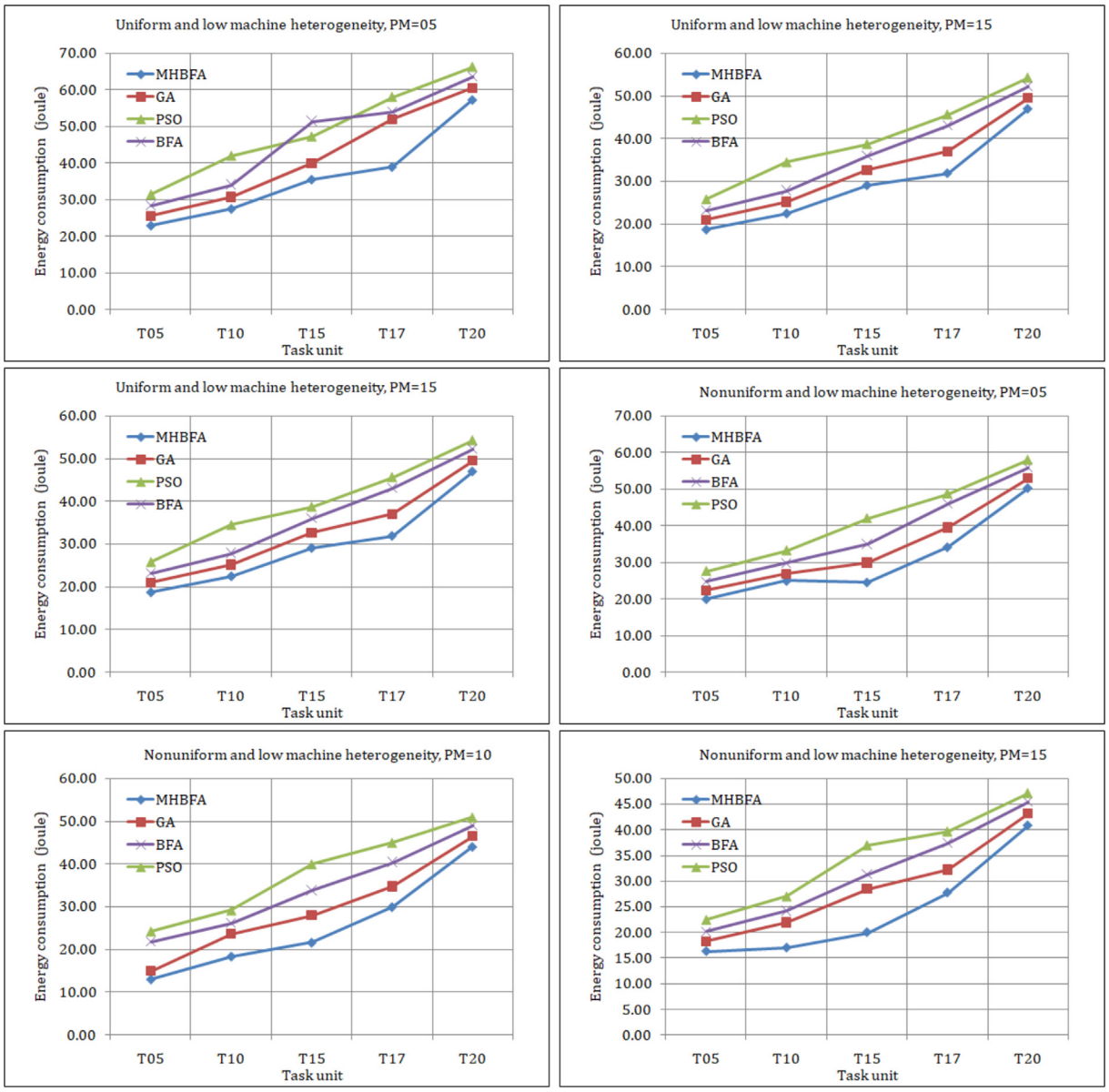

Fig. 5. Energy consumption with low resource heterogeneity.

hyper volume used with dimension above two) ratio [43,50–55]. We have presented the statistical result using [43] simulation model so as to study the statistical result analysis of the MHBFA algorithm.

The mean coverage ratio of proposed algorithm with the GA, BFA and PSO is given in Table 8. We compute the mean coverage ratio by taking the average of $C(x_i, y_j)$, $1 \leq i \leq 50$, $1 \leq j \leq 50$, where x and y represent MHBFA with the GA, BFA and PSO, and x and y is different. Each data set $C(x, y)$ has 1500 values. In case of C(MHBFA, BFA) the mean coverage ratios (Mean) (0.52256), standard deviation(0.2641) (SD) and Covariance (CV) (50.53) and in C(MHBFA, GA) (0.426, 0.2152, 50.51) are superior to the ratios C(BFA, MHBFA) (0.35358, 0.2559, 72.38) and C(BFA, GA) (0.50694 0.2543 50.16), respectively.

The Wilcoxon test focused on the algorithm, MHBFA, which had the higher mean coverage ratios. We studied the behavior of this MHBFA on the remaining algorithms (GA, BFA, and PSO). The sample data came from using the MHBFA, GA, BFA and PSO at their coverage ratios in 50 independent runs. Each data set $C(x, y)$ has 1500 values.

When we conduct the Friedman's test to obtain Chi square value we obtain 700.06 and p-value of six data sets as given in Table 9.

The p-value is lower as compared to $\alpha = 0.05$. The results obtained from the test has significance difference among them. Observation to these results says that a significant difference can be observed by applying post-hoc statistical analysis among these algorithms. The Wilcoxon z values in and corresponding p values C(MHBFA,GA) vs. C(GA,MHBFA),

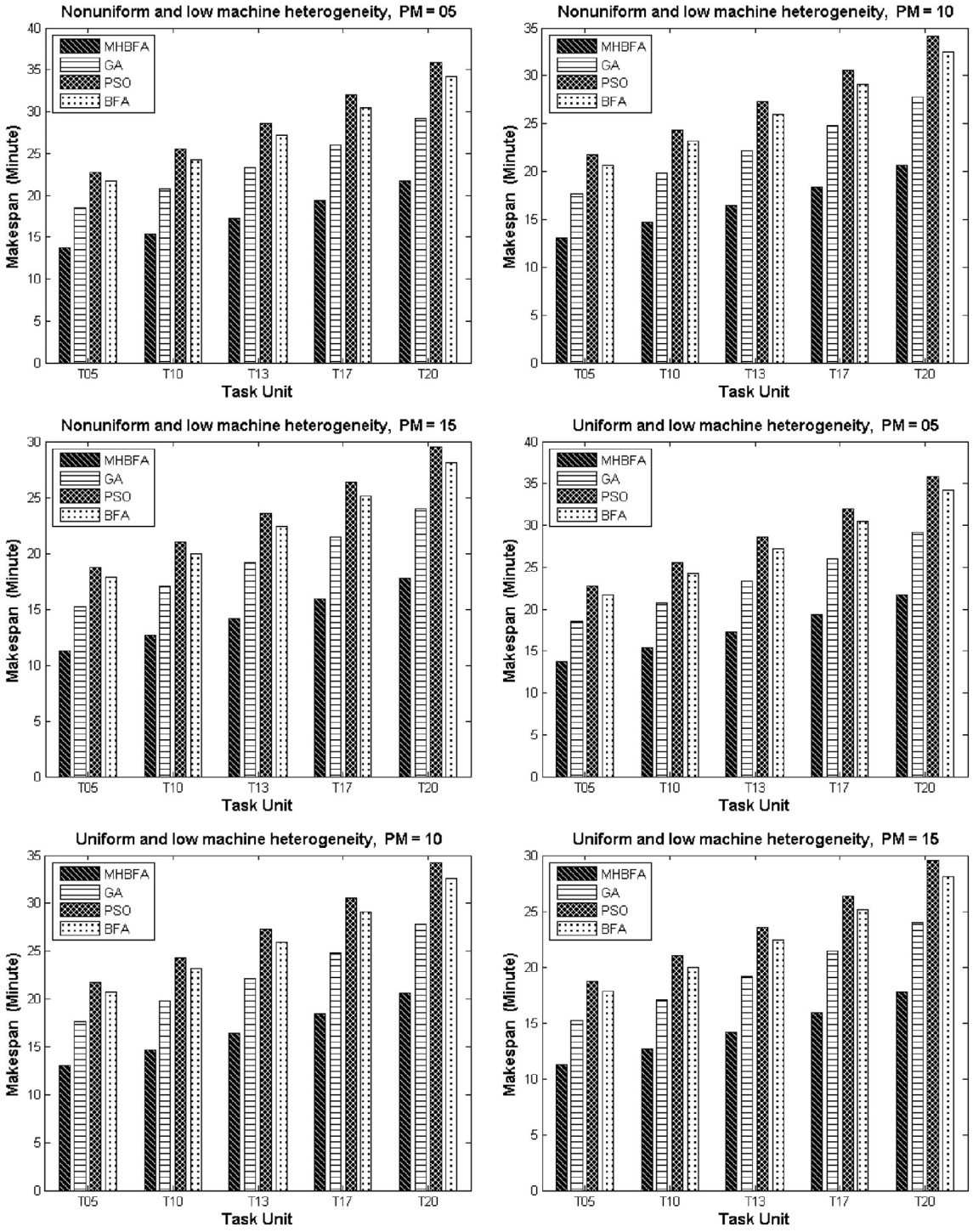

Fig. 6. Makespan with low resource heterogeneity.

C(MHBFA,BFA) vs. C(BFA,MHBFA), C(BFA,GA) vs. C(GA,BFA), C(GA,PSO) vs. C(PSO,GA), C(BFA, PSO) vs. C(PSO,BFA) and C(MHBFA,PSO) vs. C(PSO, MHBFA) are $(p<0.00001)$, $(p<0.0319)$, $(p=0.00001)$, $(p<0.2238)$, $(p=0.000001)$ and $(p<0.5507)$ as given in Table 9 for all three examples corresponds to different pair of algorithms. When we observe the z-value of the analysis results of C(MHBFA,GA) vs. C(GA,MHBFA), C(MHBFA,BFA) vs. C(BFA,MHBFA), C(BFA,GA) vs. C(GA,BFA), C(GA,PSO) vs. C(PSO,GA), C(BFA, PSO) vs. C(PSO,BFA) and C(MHBFA,PSO) vs. C(PSO, MHBFA) is 14.31, 2.145, 17.35, 1.216, 13.35 and 2.216, and these values give an impression that there is a significant difference between two algorithms because Wilcoxon z value is bigger. That is, the performance of MHBFA performs better as compared to the performances of GA, BFA, and PSO in finding the better Pareto-optimal solutions. Authors in Refs. [43,56] confirmed the use of the great nonparametric statistical tests to carry out comparisons

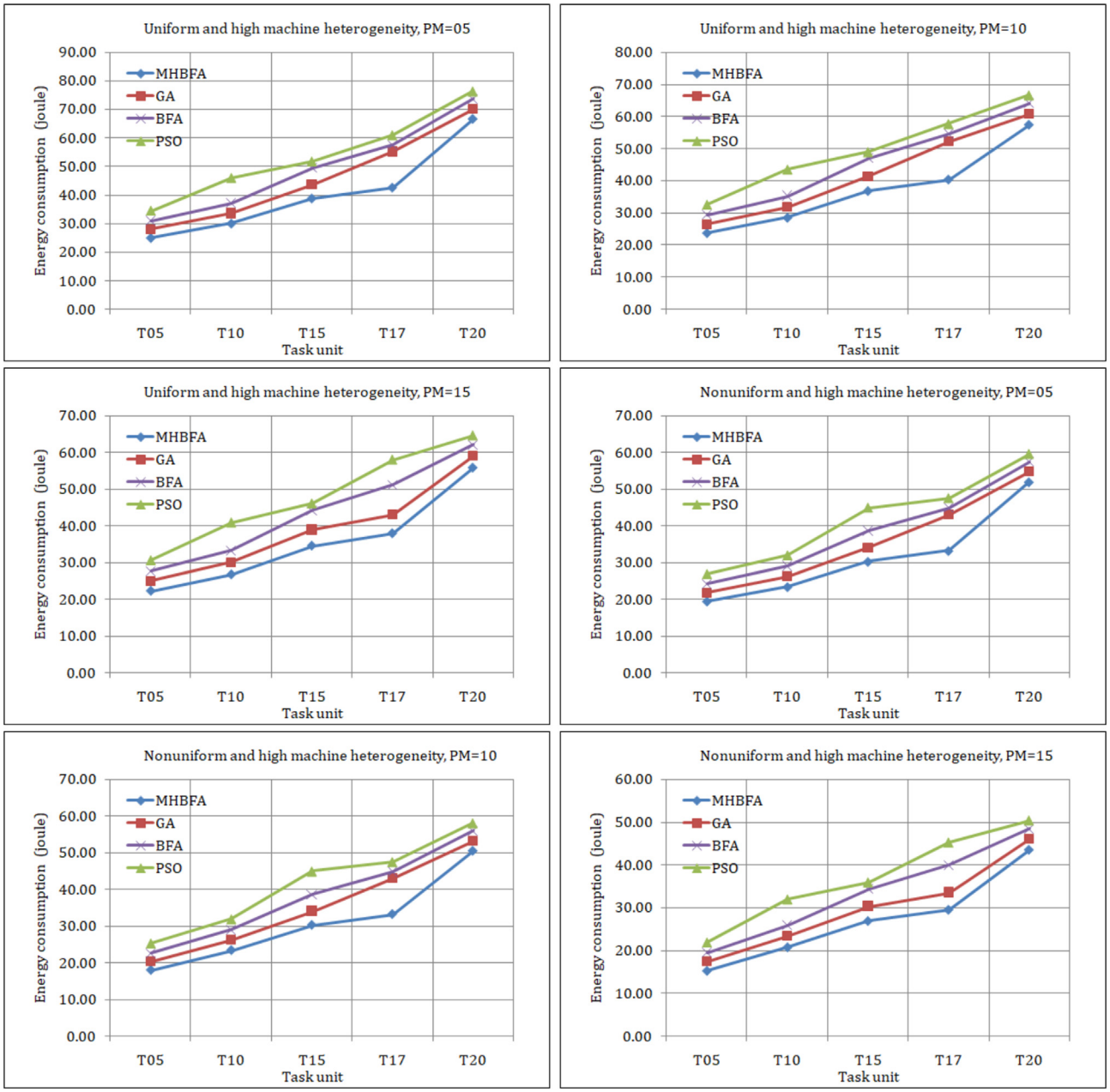

Fig. 7. Energy consumption with high resource heterogeneity.

of optimization methods. This study used Wilcoxon matched-pairs signed-rank test to compare two algorithms in the single-problem analysis [43,57].

Table 10 shows the results of performance comparisons of MHBFA with the GA, BFA, and PSO regarding SP, D, and HA in different 50 runs of the five, ten and fifteen-task unit with five, ten and fifteen PMs. There exist an inverse relation between SP and D and HA. With the increase in the value of SP the D and the HA significantly decreases. From the result we can conclude that the value of the SP, D, and HA obtained by applying the MHBFA is far superior than that of GA, BFA, and PSO. In summary, the above experimental results confirm that the MHBFA outperforms both the GA, BFA, and PSO in finding the better Pareto-optimal solutions.

The work proposed in Ref. [58] uses discrete bacteria foraging algorithm (DBFA), the problem in this technique is that it fails to achieve global optima, where in the proposed algorithm we take care both local and global optimum [30,39,40].

8.3. Scalability of the proposed MHBFA

The last set of experiments and statistical analysis is used to study whether the integrated management solution is scalable for large size of cloud environment with hundreds of task units. The scalability of our solution is based on computational complexity of MHBFA algorithm for task scheduling problem. The reason is that the process of mapping tasks to PMs is

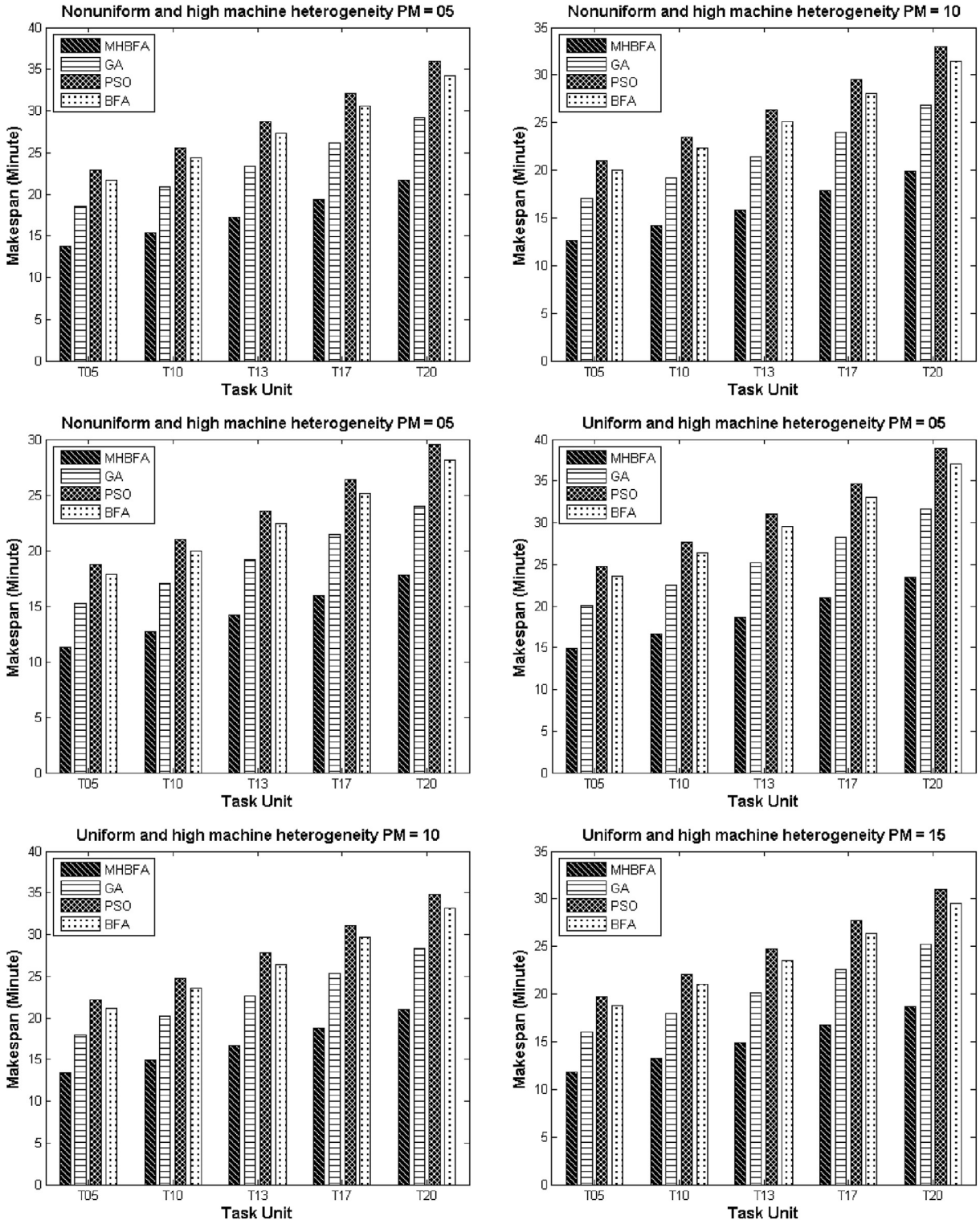

Fig. 8. Makespan with high resource heterogeneity.

known to be NP-hard. The average number of tasks executed per PMs are called the task execution ratio. The ratio is an important factor which affects the time to solve the placement problem. For this experiment, we considered three values of task execution ratio is: 2:1, 4:1 and 6:1, and the number of task executed is varied from 23 to 200. The execution time of the entire algorithm was measured on a 2.4 GHz Intel Core i3 machine. Figs. 5–8 show the result of the experiment conducted. From the figure we can see that the running time of the MHBFA algorithm increases as the task unit increases. It is because, when the task execution ratio increases, the number of PM to accommodate tasks rises up, making the placement problem harder. In addition, the statistical result shows that MHBHA performs outstanding with the increase in number of

Table 8
Results of comparisons in terms of mean coverage ratio.

	5 Task unit			10 Task unit			15 Task unit		
	Mean	SD	CV	Mean	SD	CV	Mean	SD	CV
C(MHBFA, BFA)	0.52256	0.026406	5.05	0.600944	0.032743	5.45	0.69	0.04	5.32
C(MHBFA, GA)	0.426	0.021516	5.05	0.4899	0.02668	5.45	0.56	0.03	5.32
C(BFA, MHBFA)	0.35358	0.025591	7.24	0.406617	0.031733	7.80	0.47	0.04	7.62
C(BFA, GA)	0.50694	0.025428	5.02	0.582981	0.031531	5.41	0.67	0.04	5.28
C(GA, MHBFA)	0.40896	0.02445	5.98	0.470304	0.030318	6.45	0.54	0.03	6.30
C(GA, BFA)	0.32944	0.019071	5.79	0.378856	0.023648	6.24	0.43	0.03	6.10
C(GA, PSO)	0.27974	0.026895	9.61	0.321701	0.03335	10.37	0.37	0.04	10.12
C(PSO, GA)	0.2769	0.025917	9.36	0.318435	0.032137	10.09	0.37	0.04	9.86

Table 9
Wilcoxon test in the scheduling problem.

	5 Task unit		10 Task unit		15 Task unit	
	Z-value	*P*-value	*Z*-value	*P*-value	*Z*-value	*P*-value
C(MHBFA,GA) vs. C(GA, MHBFA)	14.31	($p < 0.00001$)	17.4582	($p < 0.00019$)	18.4902	($p < 0.00013$)
C(MHBFA,BFA) vs. C(BFA, MHBFA)	2.145	($p < 0.0319$)	2.6169	($p = 0.04985$)	3.6489	($p = 0.02786$)
C(BFA, GA) vs. C(GA, BFA)	17.35	($p = 0.00001$)	21.167	($p = 0.000021$)	23.199	($p = 0.000011$)
C(GA, PSO) vs. C(PSO, GA)	1.216	($p < 0.2238$)	1.48352	($p < 0.3657$)	3.51552	($p < 0.2113$)
C(BFA, PSO) vs. C(PSO, BFA)	13.35	($p = 0.000001$)	18.167	($p = 0.000017$)	21.199	($p = 0.000012$)
C(MHBFA, PSO) vs. C(PSO, MHBFA)	2.216	($p < 0.5507$)	1.48352	($p < 0.233657$)	3.51552	($p < 0.29875$)

PM in the cloud data center. Therefore, we can say that the proposed MHBFA algorithm can be suitable for large-scale cloud data centers.

9. Conclusions and future work

In this paper, a generic task scheduling algorithm in cloud computing environment is proposed, based on bacteria foraging and genetic algorithm concept. To handle the trade-off between the makespan and energy consumption cost functions, the problem is modeled as a multi objective optimization problem. One of the most effective and simplest optimization methods, known as hybrid BFA based approach, has been applied to get Pareto optimal solutions for the given problem. Extensive simulations have been carried out to show the effectiveness and scalability of the proposed strategy on

Table 10
Results of performance comparisons of MHBFA, GA, BFA, PSO terms of SP, D, and HA.

			5 PM				10 PM				15 PM			
			MHBFA	GA	BFA	PSO	MHBFA	GA	BFA	PSO	MHBFA	GA	BFA	PSO
5 Task unit	SP	avg.	0.0458	0.0485	0.0518	0.0557	0.0347	0.0374	0.0407	0.0446	0.0287	0.0314	0.0347	0.0386
		sd	0.0375	0.0382	0.0390	0.0398	0.0264	0.0271	0.0279	0.0287	0.0204	0.0211	0.0219	0.0227
	D	avg.	0.7195	0.7052	0.6633	0.6132	0.7084	0.6941	0.6522	0.6021	0.7024	0.6881	0.6462	0.5961
		sd	0.0452	0.0457	0.0467	0.0497	0.0341	0.0346	0.0356	0.0386	0.0281	0.0286	0.0296	0.0326
	HA	avg.	0.9243	0.9036	0.8555	0.8339	0.9132	0.8925	0.8444	0.8228	0.9072	0.8865	0.8384	0.8168
		sd	0.0379	0.0377	0.0375	0.0375	0.0268	0.0266	0.0264	0.0264	0.0208	0.0206	0.0204	0.0204
10 Task unit	SP	avg.	0.0578	0.0605	0.0638	0.0677	0.0467	0.0494	0.0527	0.0566	0.0407	0.0434	0.0467	0.0506
		sd	0.0495	0.0502	0.0510	0.0518	0.0384	0.0391	0.0399	0.0407	0.0324	0.0331	0.0339	0.0347
	D	avg.	0.7315	0.7172	0.6753	0.6252	0.7204	0.7061	0.6642	0.6141	0.7144	0.7001	0.6582	0.6081
		sd	0.0572	0.0577	0.0587	0.0617	0.0461	0.0466	0.0476	0.0506	0.0401	0.0406	0.0416	0.0446
	HA	avg.	0.9363	0.9156	0.8675	0.8459	0.9252	0.9045	0.8564	0.8348	0.9192	0.8985	0.8504	0.8288
		sd	0.0499	0.0497	0.0495	0.0495	0.0388	0.0386	0.0384	0.0384	0.0328	0.0326	0.0324	0.0324
15 Task unit	SP	avg.	0.0598	0.0625	0.0658	0.0697	0.0487	0.0514	0.0547	0.0586	0.0427	0.0454	0.0487	0.0526
		sd	0.0515	0.0522	0.0530	0.0538	0.0404	0.0411	0.0419	0.0427	0.0344	0.0351	0.0359	0.0367
	D	avg.	0.7335	0.7192	0.6773	0.6272	0.7224	0.7081	0.6662	0.6161	0.7164	0.7021	0.6602	0.6101
		sd	0.0592	0.0597	0.0607	0.0637	0.0481	0.0486	0.0496	0.0526	0.0421	0.0426	0.0436	0.0466
	HA	avg.	0.9383	0.9176	0.8695	0.8479	0.9272	0.9065	0.8584	0.8368	0.9212	0.9005	0.8524	0.8308
		sd	0.0519	0.0517	0.0515	0.0515	0.0408	0.0406	0.0404	0.0404	0.0348	0.0346	0.0344	0.0344

three recent and state of the art algorithm. Simulation results show that using the proposed strategy, the convergence speed to the cost function, makespan and energy consumption have been optimized significantly. A comparative analysis with other strategies shows that the proposed strategy is comparable with multiobjective algorithm.

Possible topics for future works include: (1) Examining the role of each factor related to the similarity function, that is, task type, dependency relationship between tasks and input and output data sizes of task; (2) Although MHBFA does speed up the convergence rate, it incurs extra timing for crossover and mutation and chemotaxis and reproduction process, this process need to be further studied; (3) To further improve the algorithm's performance, it is necessary to explore the respective influence of operators and parameters in GA, develop into the mathematical theory on MOO, develop new analysis tools, and establish more effective evaluation and termination criteria; and (4) To improve the proposed method and use it in service-oriented manufacturing or industry system such as cloud manufacturing system.

References

[1] Mezmaz M, Melab N, Kessaci Y, Lee YC, Talbi E-G, Zomaya AY, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J Parallel Distr Comput 2011;71(11):1497–508.

[2] Ullman JD. *NP*-complete scheduling problems. J Comput Syst Sci 1975; 10(3):384–93.

[3] Chen Z-G, Du K-J, Zhan Z-H, Zhang J. Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In: Evolutionary computation (CEC), 2015 IEEE congress on. IEEE; 2015. p. 708–14.

[4] Shuja J, Madani SA, Bilal K, Hayat K, Khan SU, Sarwar S. Energy-efficient data centers. Computing 2012;94(12):973–94.

[5] Kim DH, Cho JH. Intelligent control of AVR system using GA-BF. In: International conference on knowledge-based and intelligent information and engineering systems. Springer; 2005. p. 854–9.

[6] Singh S, Kalra M. Scheduling of independent tasks in cloud computing using modified genetic algorithm. In: Computational intelligence and communication networks (CICN), 2014 international conference on. IEEE; 2014. p. 565–9.

[7] Singh S, Chana I. Cloud resource provisioning: survey, status and future research directions. Knowl Inform Syst 2016;49(3):1005–69.

[8] Hagras T, Janeček J. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. Parallel Comput 2005;31(7):653–70.

[9] Qiu M, Sha EH-M. Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems. ACM Trans Des Autom Electron Syst 2009;14(2):25.

[10] Germain-Renaud C, Rana OF. The convergence of clouds, grids, and autonomics. IEEE Internet Comput 2009;13(6):9.

[11] Passino KM. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Contr Syst Mag 2002;22(3):52–67.

[12] Liu Y, Passino K. Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors. J Optim Theor Appl 2002;115(3):603–28.

[13] Abraham A, Biswas A, Dasgupta S, Das S. Analysis of reproduction operator in bacterial foraging optimization algorithm. In: Evolutionary computation, 2008. CEC 2008 (IEEE world congress on computational intelligence). IEEE congress on. IEEE; 2008. p. 1476–83.

[14] Tamilselvan L, Anbazhagi, Shakkeera. Qos based dynamic task scheduling in iaas cloud. In: Recent trends in Information Technology (ICRTIT), 2014 International Conference on. IEEE; 2014. p. 1–8.

[15] Tao F, Feng Y, Zhang L, Liao TW. CLPS-GA: a case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Appl Soft Comput 2014;19:264–79.

[16] Netto MA, Buyya R. Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems. In: Parallel & distributed processing, 2009. IPDPS 2009. IEEE international symposium on. IEEE; 2009. p. 1–11.

[17] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generat Comput Syst 2012;28(5):755–68.

[18] Fernández-Baca D. Allocating modules to processors in a distributed system. IEEE Trans Software Eng 1989;15(11):1427–36.

[19] Gelenbe E, Lent R, Douratsos M. Choosing a local or remote cloud. In: Network cloud computing and applications (NCCA), 2012 second symposium on. IEEE; 2012. p. 25–30.

[20] Kong J, Choi J, Choi L, Chung SW. Low-cost application-aware DVFS for multi-core architecture. In: Convergence and hybrid information technology, 2008. ICCIT'08. Third international conference on, vol. 2. IEEE; 2008. p. 106–11.

[21] Kolpe T, Zhai A, Sapatnekar SS. Enabling improved power management in multicore processors through clustered DVFS. In: Design, automation & test in Europe conference & exhibition (DATE), 2011. IEEE; 2011. p. 1–6.

[22] Maurer M, Emeakaroha VC, Brandic I, Altmann J. Cost–benefit analysis of an SLA mapping approach for defining standardized cloud computing goods. Future Generat Comput Syst 2012;28(1):39–47.

[23] Omara FA, Arafa MM. Genetic algorithms for task scheduling problem. J Parallel Distr Comput 2010;70(1):13–22.

[24] Balin S. Non-identical parallel machine scheduling using genetic algorithm. Expert Syst Appl 2011;38(6):6814–21.

[25] Deb K. Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons; 2001.

[26] Niu B, Wang H, Wang J, Tan L. Multi-objective bacterial foraging optimization. Neurocomputing 2013;116:336–45.

[27] Chana I, Rajni. Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Generat Comput Syst 2013;29(3): 751–62.

[28] Okaeme NA, Zanchetta P. Hybrid bacterial foraging optimization strategy for automated experimental control design in electrical drives. IEEE Trans Ind Inform 2013;9(2):668–78.

[29] Nayak SK, Padhy SK, Panigrahi SP. A novel algorithm for dynamic task scheduling. Future Generat Comput Syst 2012;28(5):709–17.

[30] Kim DH, Abraham A, Cho JH. A hybrid genetic algorithm and bacterial foraging approach for global optimization. Inform Sci 2007;177(18): 3918–37.

[31] Li D, Li M, Shen Y, Wang Y, Wang Q. GA-BFO based signal reconstruction for compressive sensing. In: Information and automation (ICIA), 2013 IEEE international conference on. IEEE; 2013. p. 1023–8.

[32] Lin X, Ke S, Li Z, Weng H, Han X. A fault diagnosis method of power systems based on improved objective function and genetic algorithm-Tabu search. IEEE Trans Power Deliv 2010;25(3):1268–74.

[33] Noman N, Iba H. Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 2008;12(1):107–25.

[34] Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M. Implementation of an effective hybrid ga for large-scale traveling salesman problems. IEEE Trans Syst Man Cybernet, Part B (Cybernetics) 2007;37(1):92–9.

[35] Li S, Tan M, Tsang IW, Kwok JT-Y. A hybrid pso-bfgs strategy for global optimization of multimodal functions. IEEE Trans Syst Man Cybernetics, Part B (Cybernetics) 2011;41(4):1003–14.

[36] Zhan Z-H, Zhang J, Li Y, Chung HS-H. Adaptive particle swarm optimization. IEEE Trans Syst Man Cybernetics Part B (Cybernetics) 2009; 39(6):1362–81.

[37] Hsieh S-T, Sun T-Y, Liu C-C, Tsai S-J. Efficient population utilization strategy for particle swarm optimizer. IEEE Trans Syst Man Cybernet Part B (Cybernetics) 2009;39(2):444–56.

[38] Yen GG, Leong WF. Dynamic multiple swarms in multiobjective particle swarm optimization. IEEE Trans Syst Man Cybern Syst Hum 2009; 39(4):890–911.

[39] Bakwad KM, Pattnaik SS, Sohi B, Devi S, Panigrahi BK, Das S, et al. Hybrid bacterial foraging with parameter free PSO. In: Nature & biologically inspired computing, 2009. NaBIC 2009. World congress on. IEEE; 2009. p. 1077−81.
[40] Mishra S, Panigrahi B, Tripathy M. A hybrid adaptive-bacterial-foraging and feedback linearization scheme based D-STATCOM. In: Power system technology, 2004. PowerCon 2004. 2004 international conference on, vol. 1. IEEE; 2004. p. 275−80.
[41] Zhou Y, Hu H, Luo B. Workflow scheduling in grid based on bacterial foraging optimization. Proc Aware Syst 2015:21.
[42] Jena R. Multi objective task scheduling in cloud environment using nested PSO framework. Proc Comput Sci 2015;57:1219−27.
[43] Tsai J-T, Fang J-C, Chou J-H. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput Oper Res 2013;40(12): 3045−55.
[44] Ali S, Siegel HJ, Maheswaran M, Hensgen D, Ali S. Representing task and machine heterogeneities for heterogeneous computing systems. J Appl Sci Eng 2000;3(3):195−207.
[45] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 1999;3(4):257−71.
[46] Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 2003;7(2):117−32.
[47] Zitzler E, Knowles J, Thiele L. Quality assessment of Pareto set approximations. Multiobj Optim 2008:373−404.
[48] Coello CC, Lamont GB, Van Veldhuizen DA. Evolutionary algorithms for solving multi-objective problems. Springer Science & Business Media; 2007.
[49] Okabe T, Jin Y, Sendhoff B. A critical survey of performance indices for multi-objective optimisation. In: Evolutionary computation, 2003. CEC'03. The 2003 congress on, vol. 2. IEEE; 2003. p. 878−85.
[50] Bandyopadhyay S, Pal SK, Aruna B. Multiobjective gas, quantitative indices, and pattern classification. IEEE Trans Syst Man Cybernetics, Part B (Cybernetics) 2004;34(5):2088−99.
[51] Araújo DR, Bastos-Filho CJ, Barboza EA, Chaves DA, Martins-Filho JF. A performance comparison of multi-objective optimization evolutionary algorithms for all-optical networks design. In: Computational intelligence in multicriteria decision-making (MDCM), 2011 IEEE symposium on. IEEE; 2011. p. 89−96.
[52] Hsu C-H, Tsou C-S, Yu F-J. Multicriteria tradeoffs in inventory control using memetic particle swarm optimization. Int J Innov Comput Inform Control 2009;5(11):3755−68.
[53] Reddy MJ, Kumar DN. Multiobjective differential evolution with application to reservoir system optimization. J Comput Civ Eng 2007;21(2):136−46.
[54] Santana-Quintero LV, Coello CAC. An algorithm based on differential evolution for multi-objective problems. Int J Comput Intell Res 2005; 1(1):151−69.
[55] Tsou C-S, Chang S-C, Lai P-W. Using crowding distance to improve multi-objective PSO with local search. In: Swarm intelligence, focus on ant and particle swarm optimization. InTech; 2007.
[56] García S, Molina D, Lozano M, Herrera F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC' 2005 special session on real parameter optimization. J Heuristics 2009;15(6):617−44.
[57] Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bull 1945;1(6):80−3.
[58] Liu C, Wang J, Leung JY-T, Li K. Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm. Int J Prod Res 2016;54(3):923−44.

Srichandan Sobhanayak is currently working as an Assistant professor in Department of Computer Science and Engineering, International Institute of Information Technology (IIIT), Bhubaneswar, Odisha, India. His current research interests include Internet of Things, Cloud and Grid Computing, Wireless Network, Network Security, Mobile Computing, Embedded System and Distributed System. He has published more than 15 papers in international journals and conferences in these areas.

Ashok Kumar Turuk is currently working as an Associate Professor in Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Odisha, India. He received his PhD in Computer Science and Engineering in the year 2005 from the Indian Institute of Technology, Kharagpur, India, and BTech and MTech in Computer Science and Engineering in the year 1992 and 2000 respectively from National Institute of Technology, India. His current research interests include Optical Network, Wireless Network, Network Security, Mobile Computing, Embedded System and Distributed System. He has published more than 50 papers in international journals and conferences in these areas.

Bibhudatta Sahoo is an Assistant Professor in the Department of Computer Science and Engineering at the National Institute of Technology, Rourkela, India where he has been a faculty member since 2000. He is a member of the Communication and Computing Research Group, and Professor in charge of Cloud Computing Research Laboratory. He received his Ph.D. Degree in computer science from National Institute of Technology, Rourkela for his dissertation on ”Dynamic load balancing strategies in heterogeneous distributed system”. His research interests lie in the area of Parallel and Distributed Systems, Cloud Computing, Sensor Network, Algorithms for VLSI Design, Internet of Things, Software defined networks, Multicore Architecture, 5g Networks and Algorithmic Engineering. Dr. Sahoo is a Life Member of the Institution of Electronics and Telecommunication Engineers (IETE), the Computer Society of India (CSI), the Indian Society for Technical Education (ISTE), the Indian Science Congress Association (ISCA), and the Orissa Science Academy. Dr. Sahoo is also a member of IEEE, and professional member of ACM, and the author or co-author of over 100 publications and book chapters. For more information, visit https://scholar.google.co.in/citations?hl=en&user=ffg_LoYAAAAJ.