



Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing

Maha Zeedan¹ · Gamal Attiya¹ · Nawal El-Fishawy¹

Received: 7 September 2020 / Accepted: 8 September 2022 / Published online: 1 October 2022
© The Author(s) 2022

Abstract

This paper presents a hybrid approach based Binary Artificial Bee Colony (BABC) and Pareto Dominance strategy for scheduling workflow applications considering different Quality of Services (QoS) requirements in cloud computing. The main purpose is to schedule a given application onto the available machines in the cloud environment with minimum makespan (i.e. schedule length) and processing cost while maximizing resource utilization without violating Service Level Agreement (SLA) among users and cloud providers. The proposed approach is called Enhanced Binary Artificial Bee Colony based Pareto Front (EBABC-PF). Our proposed approach starts by listing the tasks according to priority defined by Heterogeneous Earliest Finish Time (HEFT) algorithm, then gets an initial solution by applying Greedy Randomized Adaptive Search Procedure (GRASP) and finally schedules tasks onto machines by applying Enhanced Binary Artificial Bee Colony (BABC). Further, several modifications are considered with BABC to improve the local searching process by applying circular shift operator then mutation operator on the food sources of the population considering the improvement rate. The proposed approach is simulated and implemented in the WorkflowSim which extends the existing CloudSim tool. The performance of the proposed approach is compared with Heterogeneous Earliest Finish Time (HEFT) algorithm, Deadline Heterogeneous Earliest Finish Time (DHEFT), Non-dominated Sort Genetic Algorithm (NSGA-II) and standard Binary Artificial Bee Colony (BABC) algorithm using different sizes of tasks and various benchmark workflows. The results clearly demonstrate the efficiency of the proposed approach in terms of makespan, processing cost and resources utilization.

✉ Maha Zeedan
m_zeedan@yahoo.com

Gamal Attiya
gamal.atiya@yahoo.com

Nawal El-Fishawy
nelfishawy@hotmail.com

¹ Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University, Shibin el Kom, Egypt

Keywords Cloud computing · Workflow · Scheduling · Algorithms · Artificial Bee Colony · Multi-objective optimization

1 Introduction

Cloud computing is a distributed heterogeneous computing model providing many services through the Internet without the violation of Service level agreement (SLA). The cloud services are provided as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The cloud computing power is supplied by a collection of data centers (DCs) that are typically installed with massive hosts (physical servers). These hosts are transparently managed by the virtualization services that allow sharing their capacities among virtual instances of servers (VMS) [1].

Generally, scientific workflows in domains such as astronomy and biology can be modeled as composed of large number of smaller subprocesses (i.e., tasks) to be processed or managed. Processing such large amounts of data requires the use of a distributed collection of computation and storage facilities as in the cloud. There is no single solution for such problems but a set of alternatives with different trade-offs among objectives [2]. In cloud environment, the data centers have unlimited resources therefore there is a need to schedule workflow applications for execution based on certain criteria such as makespan (i.e., overall execution time of all the tasks), cost, budget, reliability, deadline, and resource utilization [3]. A workflow scheduling problem is a multi-objective optimization problem that has trade-off objectives which means that none of the objective functions can be improved in value without degrading some of the other objective values [4].

Recently, several meta-heuristic algorithms are the most common methods to solve multi-objective task scheduling problem. The common feature among such algorithms is the way of their search that depends on the exploration as the process of visiting a new search space and the exploitation as making use of these search space regions. These algorithms can be divided into two categories a single-based and a population-based meta-heuristics [5]. A Greedy Randomized Adaptive Search Procedure (GRASP) is an example of single-based meta-heuristic technique include local search [6]. Population-based meta-heuristic techniques include genetic algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Optimization (ABC) [7].

Artificial bee colony (ABC) is proposed by Karaboga in [8] as one of the swarm intelligence-based algorithms that simulates the foraging behavior of bees. ABC has several advantages which include easy to implement, flexible and robust. It is implemented with only three control parameters which are colony size, limit, and the maximum number of cycles. Due to these advantages, it has been successfully tailored for the different optimization problems such as workflow scheduling problem [9].

This paper tackles the multi-objective workflow scheduling problem in cloud computing using a new efficient hybrid approach called EBABC-PF. The purpose is to

minimize the makespan and the processing cost of a given application while maximize the resource utilization based on the workload. The proposed hybrid approach presents a new efficient multi-phase hybrid approach by combining the advantages of several meta-heuristic techniques. It starts by listing the given tasks of an application according to their priorities using Heterogeneous Earliest Finish Time (HEFT) [10, 11] algorithm. It then gets an initial population by applying greedy randomized search algorithm (GRASP) to overcome the random initialization of the food sources (feasible solutions) in the population during searching process and so achieving acceptable convergence and diversity among food sources (feasible solutions) in the search space. Further, the tasks are scheduled onto virtual machines by applying binary artificial bee colony algorithm (ABC) [12, 13] with some improvements within the local searching process by applying the circular shift then the mutation operators among the virtual machines of maximum completion time and minimum completion time on the food sources of the population. Our proposed approach can achieve minimum makespan and total processing cost with load balancing among the virtual machines and so the resource utilization can also be maximized. Finally, as a set of solutions with trade-off among objectives are generated, the non-dominance concept is applied for ranking the feasible solutions to get the best near-optimal solution for workflow scheduling in cloud.

Our main research contributions in this paper are as follows:

1. This paper suggests the task scheduling in cloud with an Enhanced Binary Artificial Bee Colony based Pareto Front (EBABC-PF) to solve the workflow scheduling as a multi-objective optimization problem.
2. The proposed hybrid approach EBABC-PF regards the performance metrics: the makespan, the processing cost and the resource utilization to solve task scheduling problem in cloud.
3. To demonstrate the efficiency of EBABC-PF algorithm, it is coded in Java and embedded into WorkflowSim simulator based Cloudsim simulator [14, 15] that simulates large scale cloud computing infrastructure with five groups of practical workflows benchmarks, i.e., Montage, CyberShake, Epigenomics, LIGO, and SIPHT.

The remainder of this paper is organized as follows: Sect. 2 presents a survey of related work. Section 3 describes the scheduling process and formulates the scheduling problem as a multi-objective optimization problem. Section 4 presents the proposed hybrid approach in details and its complexity analysis while Sect. 5 presents the experimental results and discussion. Finally, Sect. 6 presents the concluding remarks and future work of this research paper.

2 Related work

Several meta-heuristic scheduling algorithms for task scheduling in cloud computing have been proposed. They gained huge popularity due to its effectiveness to solve complex problems. An Improved Ant Colony Multi-Objective Optimization algorithm in [16] is suggested for optimizing makespan, cost, deadline violation rate, and resource

utilization. In [17], the authors suggest task scheduling algorithm called HABC based on Artificial Bee Colony Algorithm (ABC) to minimize makespan considering load balancing.

Currently, many scheduling algorithms are suggested for energy efficiency issue. An Improved Grouping Genetic Algorithm (IGGA) based on a greedy heuristic and a swap operation is introduced in [18] for maximal saved power by optimizing the consolidation score function based on a migration cost function and an upper bound estimation function. Multi-objective algorithms for task scheduling based on Non-dominated Sorting Genetic Algorithm (NSGA-II) [19] are suggested. The authors in [20] incorporate Dynamic Voltage Frequency Scaling System with NSGA-II for minimizing energy consumption and makespan. The authors in [21] propose a new hybrid multi-objective algorithm for task scheduling based on NSGA-II and Gravitational Search Algorithm (GSA) for minimizing response time, and execution cost while maximizing resource utilization.

Recently, multi-objective scheduling algorithms for scheduling scientific workflow applications are proposed based on meta-heuristic scheduling algorithms. The authors propose a hybrid algorithm combining Genetic Algorithm (GA) with Artificial Bee Colony Optimization (ABCO) Algorithm in [22] for workflow scheduling optimization to optimize makespan and cost simultaneously. In [23], the authors suggest an energy-efficient dynamic scheduling scheme (EDS) of real-time tasks in cloud. The suggested algorithm classifies the heterogeneous tasks and virtual machines based on a historical scheduling record. Then, merging the similar type of tasks to schedule them to maximally utilize the hosts considering energy efficiencies and optimal operating frequencies of physical hosts.

Further, task scheduling algorithms considering load balancing are proposed. In [24], task scheduling based on Artificial Bee Colony (ABC) is suggested. In [25], HABC_LJF algorithm is suggested based on Artificial Bee Colony and largest job first for minimizing makespan. The experimental results prove that the suggested algorithm outperforms those with ACO, PSO, and IPSO. A multi-objective scheduling algorithms based on particle swarm optimization (PSO) integrated with Fuzzy resource utilization (FR-MOS) is proposed in [26] for minimizing cost and makespan while considering reliability constraint, task execution location and data transportation order. In [27], a task scheduling considering deadlines, data locality and resource utilization is proposed to save energy costs and optimize resource utilization using fuzzy logic to get the available number of slots from their rack-local servers, cluster-local servers, and remote servers. In [28], a simulated-annealing-based bi-objective differential evolution (SBDE) algorithm is designed to obtain a pareto optimal set for distributed green data centers (DGDCs) to maximize the profit of the providers and minimize the average task loss possibility. In [29], the authors suggest a Non-Dominated Sorting-Based Hybrid Particle-Swarm Optimization (HPSO) algorithm to optimize both execution time and cost under deadline and budget constraints. A scheduling algorithm called energy-makespan multi-objective optimization (EM-MOO) is proposed in [30] to find a trade-off between the reducing energy consumption and makespan. The researchers in [31] design resource prediction-based scheduling (RPS) approach which maps the tasks of scientific application with the optimal virtual machines by combining the features

of swarm intelligence and multi-criteria decision-making approach. The proposed approach tries to minimize the execution time, cost.

Although several meta-heuristics strategies are presented, Artificial Bee Colony (ABC) algorithm proposed by Karaboga [32] is used in our proposed hybrid approach. ABC algorithm is an optimization algorithm based on a particular intelligent behavior of honeybee swarms. The main advantages of ABC algorithm over other optimization algorithms are exploration, exploitation, robustness, simplicity, few control parameters, fast convergence, high flexibility [33, 34].

In our proposed hybrid approach for scheduling the workflow applications, Artificial Bee Colony (ABC) algorithm as a meta-heuristics algorithm is integrated with Greedy Randomized Adaptive Search Procedure (GRASP) to overcome the random initialization of the population food sources and so achieving convergence and diversity in the search space. Then, some improvements are implemented in the local search process within each available food source by considering the load of every virtual machine. After that, the tasks are swapped between the virtual machine of the maximum completion time and the virtual machine of the minimum completion time. Our proposed approach can overcome various challenges related to multi-objective optimization of scheduling workflow applications.

3 Problem formulation and modeling

This section formulates a complete scheduling model for our proposed architecture by defining the system model, the workflow model, the proposed scheduling problem with the constraints.

3.1 System model

Workflows have been frequently used to model large scale scientific applications that demand a high-performance computing environment in order to be executed in a reasonable amount of time. These workflows are commonly modeled as a set of tasks interconnected via data or computing dependencies. The execution of workflow applications in cloud is done via a cloud workflow management system (CWfMS). Workflow management systems are responsible for managing and executing these workflows. Workflow Management System schedules workflow tasks to remote resources based on user-specified QoS requirements and SLA_based negotiation with remote resources capable of meeting those demands. The data management component of the workflow engine handles the movement and storage of data as required. There are two main stages when planning the execution of a workflow in a cloud environment. The first one is the resource provisioning phase; during this stage, the computing resources that will be used to run the tasks are selected and provisioned. In the second stage, a schedule is generated, and each task is mapped onto the best-suited resource [35].

Our proposed architecture in Fig. 1 presents a high-level architectural view of a Workflow Management System (WFMS) utilizing cloud resources to drive the

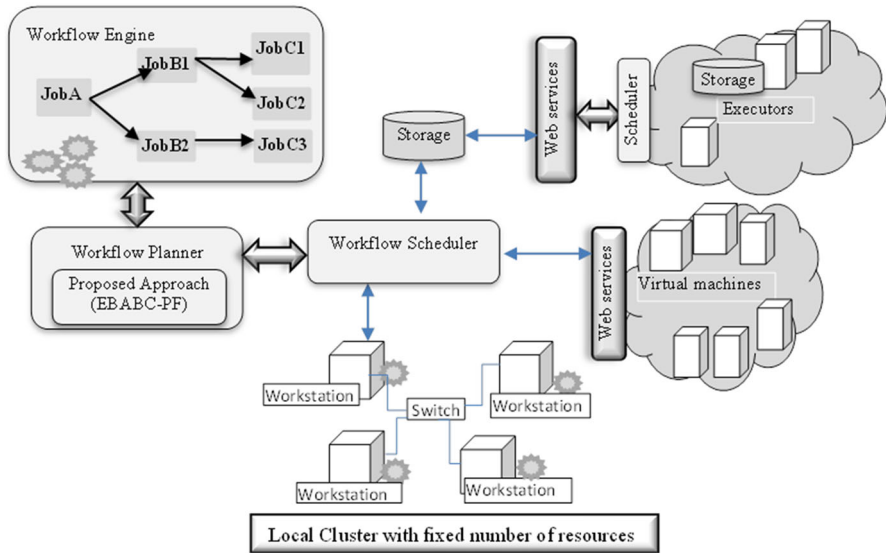


Fig. 1 The proposed model using EBABC-PF algorithm for workflow scheduling

execution of a scientific workflow application. It consists of three major parts: Workflow Planner, Workflow Engine, Clustering Engine, and Workflow Scheduler. In our proposed architecture, our proposed algorithm (EBABC-PF) is implemented in the workflow Planner for identifying suitable cloud service providers to the users then keeping track of the load on the data centers for allocation of resources that meets the Quality of Service (QoS) needs. The performance evaluation of the workflow optimization algorithms in real infrastructure is complex and time consuming. Therefore, we use WorkflowSim toolkit in our simulation-based study to evaluate these workflow systems [15].

Our proposed approach considers fundamental features of Infrastructure as a Service (IaaS) providers such as the dynamic provisioning and heterogeneity of unlimited computing resources. The Workflow Management System architecture allows end users to work with workflow composition, workflow execution planning, submission, and monitoring. These features are delivered through a Web portal or through a stand-alone application that is installed at the user's end. Scheduling dependent tasks in the workflow application is usually called static scheduling algorithm or planning algorithm because you set the mapping relation between VMs and tasks in the Workflow Planner and should not change in Workflow Scheduler.

3.2 Workflow model

This section has introduced different scientific workflows such as Montage, Cyber-Shake, Epigenomics, LIGO Inspiral, and SIPHT. The full description of these

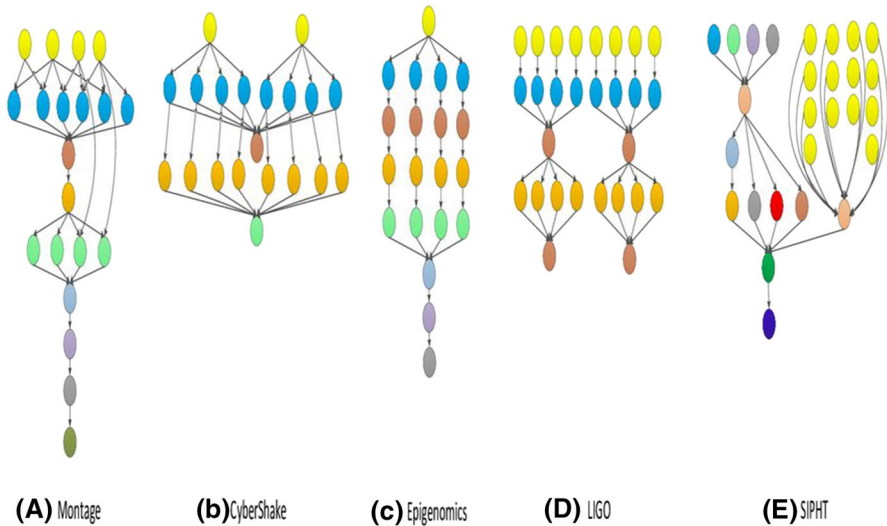


Fig. 2 Structure of the four different workflows used in the experiments. (A) Montage. (B) CyberShake. (B) Epigenomics. (D) LIGO. (E) SIPHT

workflows is presented by Juve et al. [36, 37]. Each of these workflows has different structures as seen in Fig. 2 and different data and computational characteristics. Montage workflow is an astronomy application used to generate custom mosaics of the sky based on a set of input images. Most of its tasks are characterized by being I/O intensive while not requiring much CPU processing capacity. CyberShake is used to characterize earthquake hazards by generating synthetic seismograms and may be classified as a data intensive workflow with large memory and CPU requirements. Epigenomics workflow is used in bioinformatics essentially in a data processing to automate the execution of various genome sequencing operations or tasks. LIGO Inspirational workflow is used in the physics field with the aim of detecting gravitational waves. This workflow is characterized by having CPU intensive tasks that consume large memory. Finally, SIPHT is used in bioinformatics to automate the process of searching for sRNA encoding-genes for all bacterial replicons in the National Center for Biotechnology Information database. Most of the tasks in this workflow have a high CPU and low I/O utilization.

3.3 Definitions

Cloud computing providers have several data centers at different geographical locations providing many services through the Internet without violation Service level agreement (SLA). All the computational resources are in the form of virtual machines (VMS) with different types and characteristics deployed in the data centers. They have different numbers of CPUs of different cycle times (Millions of Instructions Per Second (MIPS)), processing cores (single-core and multi-core), memory capacity and network bandwidths. Service level agreement (SLA) must be ideally set up between

customers and cloud computing providers to act as warranty. Service level agreement (SLA) specifies the details of the service to be provided and the QoS parameters such as availability, reliability, and throughput and ensures that they are delivered to the applications. Metrics must be agreed upon by all parties, and there are penalties for violating the expectations. The factor number one of whether the multi-tasks applications will run smoothly on the virtual machines is the number of cores of CPU the users need for running. Clock speed of your cores is the other factor. The elasticity of the application should be contracted and formalized as part of SLA capacity availability between the cloud provider and service owner. For example, if you want to run multiple applications at once or more resource intensive programs, the machine needs multiple CPU cores. In many cases, resources allocation decisions are application-specific and are being driven by the application-level metrics. In our experiments use WorkflowSim simulator which is based on CloudSim simulator that supports modeling of the aforementioned SLA violation scenarios. Moreover, it is possible to define particular SLA-aware policies describing how the available capacity is distributed among VMs. The number of resources that was requested but not allocated can be accounted for by CloudSim [38, 39].

We assume there is a collection of interdependent tasks of a workflow application which need to be executed in the correct sequential order on a number of heterogeneous virtual machines (VMS). A workflow application is modelled by a Directed Acyclic Graph (DAG), defined by a tuple $G(T, E)$, where T is the set of n tasks $T = \{t_1, t_2, \dots, t_n\}$ and E is a set of e edges, represent the precedence constraints. There is a list of available m virtual machines $VMS = \{vm_1, vm_2, \dots, vm_m\}$.

Definition 1 (Tasks) The task can be represented as a tuple $t_j = \{j, M, Pe\}$, where j represents the identifier of t_j , M represents the length of t_j in million instructions (MI) and Pe represents the number of processors for running a task on the virtual machine (vm).

Definition 2 (Virtual Machines) The virtual machine can be described as $vm_i = \{i, Mp, Pe\}$ where i represents identifier of vm_i , Mp is the processing speed measured in Million Instructions per Second (MIPS) per processing element at vm_i , and Pe represents the number of processing elements in a vm_j .

Definition 3 (Multi-objective optimization) The concept of dominance and Pareto optimality [5] is used to get Pareto optimal solutions. Briefly, a general formulation for a multi-objective optimization problem that has number of objective functions z ($z \geq 2$) with set S of feasible decision variables, is thus as Eq. (1):

$$PF^*(y) = \left\{ \begin{array}{l} \min F(y) = (f_1(y), f_2(y), \dots, f_z(y)) \\ \text{s.t.} \\ y \in S \vee S = \{y_1, y_2, y_3, \dots, y_d\} \end{array} \right\} \quad (1)$$

If an objective function is intended to be maximized, it is equivalent to minimize the negative of the function.

A solution is called nondominated, Pareto optimal, or Pareto efficient if none of the objective functions can be improved in value without degrading some of the other

objective values. When there are multiple objectives ($F(y)$), a feasible solution $y_1 \in S$ is said to Pareto dominate another solution $y_2 \in S$ if y_1 is not worse than y_2 in all objectives and better than y_2 in at least one objective. For minimization problem, Eq. (2) and Eq. (3) should be satisfied for Pareto dominate:

$$1. f_\beta(y_1) \leq f_\beta(y_2) \text{ for all indices } \beta \in \{1, 2, \dots, z\} \quad (2)$$

$$2. f_\eta(y_1) < f_\eta(y_2) \text{ for at least one index } \eta \in \{1, 2, \dots, z\} \quad (3)$$

3.4 Task scheduling problem modeling

In our proposed approach, both resource provisioning and scheduling are merged and modeled as an optimization problem. Our proposed task scheduling problem may be formulated as a mathematical model consists of trade-off objective functions representing the main goal(s) of the scheduling and a set of constraints representing the tasks requirements and resources availability. For simplicity, consider the following assumptions:

- 1. The task execution is a non-preemptive, i.e., the assigned task will occupy the virtual machine until finishing.
- 2. A task should not start unless all predecessor (parents) tasks are completed.
- 3. Each task has different execution time on different virtual machines (VMS) due to the cloud heterogeneity.

We assume that task execution matrix ($TA_{m \times n}$) is constructed of $m \times n$ $ET(vm_i, t_j)$ using m of virtual machines VMS and n of tasks T using Eq. (4):

$$TA_{m \times n} = \left[\begin{array}{cccc} ET_{1,1} & ET_{1,2} & \dots & ET_{1,j} \dots ET_{1,n} \\ \dots & \dots & \dots & \dots \\ ET_{i,1} & ET_{i,2} & \dots & ET_{i,j} \dots ET_{i,n} \\ \dots & \dots & \dots & \dots \\ ET_{m,1} & ET_{m,2} & \dots & ET_{m,j} \dots ET_{m,n} \end{array} \right] \quad (4)$$

In the above matrix $TA_{m \times n}$, each row represents the execution time of different tasks processed on a targeted vm , and each column represents the execution time of a task on different virtual machines. Let ET_{ij} be the execution time for task t_j corresponding to vm_i . ET_{ij} is calculated using Eq. (5):

$$ET_{ij} = \frac{M_j}{M_{p_i} \times P_{e_i}} \quad (5)$$

where: M_j is the size of task t_j ; M_{p_i} is the speed of vm_i . P_{e_i} is the number of processing elements.

The data D_{kj}^{out} represents the amount of transmitted data from the virtual machine $vm(t_k)$ that executes the task t_k to the virtual machine $vm(t_j)$ that executes the task t_j .

Let Ω_{kj} be the bandwidth between the virtual machine vm_k and the virtual machine vm_j measured in Bits per Second (B/S). The transfer time TT_{kj} between two virtual machines executing different tasks (t_k, t_j) is determined using Eq. (6):

$$TT_{ij}(t_k, t_j) = \frac{D^{out}[k, j]}{\Omega[k, j]} \quad (6)$$

Note that the transfer time between two tasks running on the same vm equals 0. The Earliest Start Time $EST(t_j)$ of task t_j is calculated using Eq. (7):

$$EST_j = \max_{t_k \in \text{pred}(t_j)} (EFT_k + TT(t_k, t_j)) \quad (7)$$

Where $\text{pred}(t_j)$ is the set of predecessors of task t_j .

The Earliest Finish Time $EFT_j(t_j)$ of task t_j is calculated as in Eq. (8):

$$EFT_j = EST_j + \min_{vm_i} \{ET_{ij}(vm_i, t_j)\} \quad (8)$$

Note that $EST_j(t_{entry}) = 0$ and $EFT_j(t_{entry}) = ET_j(vm_i, t_{entry})$.

It is necessary to define decision variable (x_{ij}) before presenting the objective functions. let x_{ij} be a binary variable given by Eq. (9):

$$x_{ij} = \begin{cases} 1 & \text{if task } t_j \text{ assigned to } vm_i \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The completion time (CT) of tasks that assigned onto a virtual machine (vm_i) is calculated as Eq. (10):

$$CT_i = \sum_{j=1}^n EFT_j x_{ij} \text{ for } j = 1, 2, 3 \dots n \quad (10)$$

The makespan (MS) is the maximum completion time of the overall virtual machines in the schedule as shown in Eq. (11):

$$MS = \max_{i \in \{1 \dots m\}} (CT_i) \text{ for } i = 1, 2, 3 \dots m \quad (11)$$

The average cloud execution time (ACT) is calculated using Eq. (12):

$$ACT = \frac{\sum_{i=1}^m CT_i}{m} \quad (12)$$

The resource utilization ($A\hat{u}$) is calculated using Eq. (13):

$$A\hat{u} = \frac{ACT}{MS} \quad (13)$$

For each vm_i , C_i^{exe} is the data processing cost per hour. For each cloud provider, the total processing cost (TC^{exe}) can be calculated using Eq. (14):

$$TC^{exe} = \sum_{i=1}^m C_i^{exe} \times CT_i \quad (14)$$

To solve the proposed scheduling problem based on the concept of dominance, the multi-objective function ($F(y)$) can be formulated to minimize makespan (MS) and total processing cost (TC^{exe}) while maximizing resource utilization ($A\hat{u}$) as Eq. (15). The multi-objective function (F) satisfying the constraints in Eqs. (16–18) can be modeled as:

$$\min(F) = \min(\text{MS}, TC^{exe}, (-A\hat{u})) \quad (15)$$

s.t.

$$\sum_{k=1}^m x_{ik} = 1 \forall task_k \quad (16)$$

$$\sum_{i=1}^n a_{ik} x_{ik} \leq b_i \forall vm_i \quad (17)$$

$$TC^{exe} \leq \text{Budget}(\text{Application}) \quad (18)$$

The constraints are formulated to meet tasks requirements and cloud resources availability. The first constraint, Eq. (16), assures that each task (t_i) is submitted to only one of the virtual machines (VMS). The second constraint, Eq. (17), guarantees that the required resources for all tasks assigned to virtual machine (vm_k) doesn't exceed the processing power of that vm_k . The third constraint, Eq. (18), ensures that the total processing cost must be less or equal to the budget dedicated to that workflow application.

4 Enhanced binary artificial bee colony based pareto front approach

This section presents a new hybrid approach EBABC-PF for solving multi-objective task scheduling of workflow application. Figure 3 presents the flowchart of the proposed hybrid approach (*EBABC-PF*). The proposed hybrid approach composed of multi-phases: priority list phase, initialization phase and allocation phase.

In the priority list phase, the Heterogeneous Earliest Finish Time (*HEFT*) algorithm [10, 11] is used to build a priority list of the submitted tasks to be scheduling. In the initialization phase, a greedy randomized adaptive search procedure (*GRASP*) [40, 41] is used to satisfy the convergence and to get feasible solutions for population. In the allocation phase, the Binary Artificial Bee Colony algorithm (*BABC*) [12, 13] is used to schedule tasks onto virtual machines.

In the proposed approach, several modifications are considered with the BABC where Right Circular Shift [42] and Mutation Operator [43] are used for producing

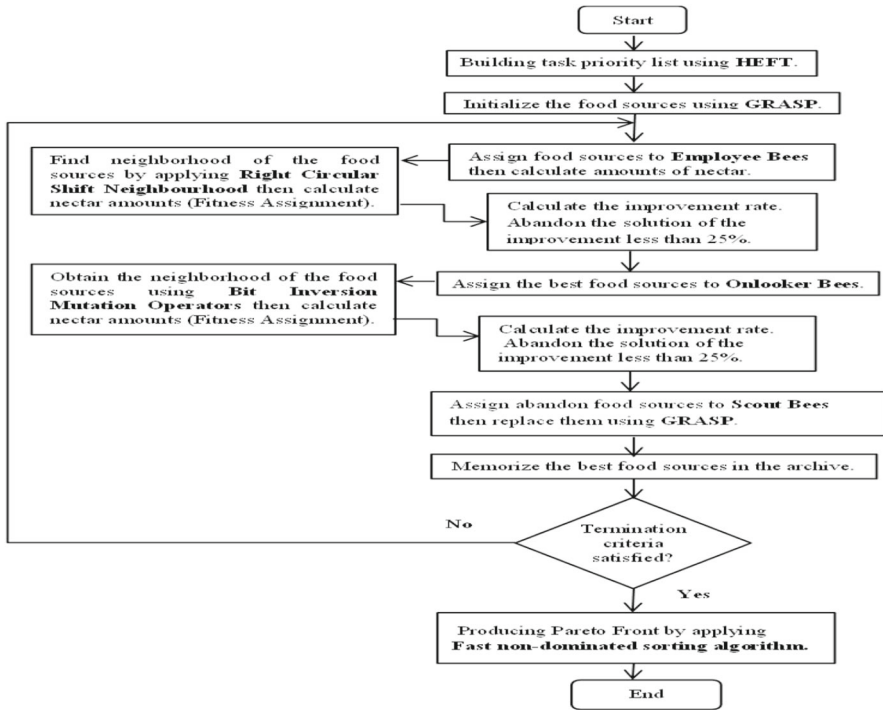


Fig. 3 Flowchart of the proposed hybrid approach EBABC-PF

the new food source (solution) to satisfy the diversity in the proposed solution space. Further, a non-dominated sorting algorithm as in [20] is used for sorting onlooker population (final solutions) based on the dominance approach and the pareto-based strategy.

4.1 Priority list phase

In this phase, Heterogeneous Earliest Finish Time (HEFT) algorithm [10, 11] is used to build a priority list of tasks. Algorithm 1 shows the pseudo-code of the HEFT ranking algorithm for task sorting phase. In this stage, tasks are sorted in descending order based on their rank value. The rank value of a task ($rank(t_i)$) is calculated using Eq. (19):

$$rank(t_j) = AVG(ET_{jk}) + \text{MAX}_{i \in succ(t_j)} (TT_{ij} + rank(t_i)) \forall k \in vms \quad (19)$$

where:

$AVG(ET_{jk})$: is the average execution time of the task j on all virtual machines.

TT_{ij} : is the transfer time between tasks t_i and t_j .

$succ(t_j)$: is the immediate successors of task t_j .

$rank(t_i)$: is the rank of the immediate successors of task t_j

Note that the transfer time TT_{ij} between two tasks running on the same vm equals 0.

Algorithm 1: Task priority phase using HEFT algorithm.

1. For each task t_j in the DAG.
 - 1.1. Calculate average execution time on all VMS ($AVG(ET_{ik})$)
 - 1.2. If task t_j is the leave node
 - 1.2.1. $rank(t_j) = AVG(ET_{jk})$
 - 1.3. else
 - 1.3.1. $rank(t_j) = AVG(ET_{jk}) + \max_{i \in succ(t_j)} (TT_{ij} + rank(t_i))$
 - 1.4. End if.
2. End for
3. Sort the tasks in a scheduling list in descending order using rank values.

4.2 Initialization phase

For generating initial populations (feasible solutions) in the proposed hybrid approach EBABC-PF, Greedy Randomized Adaptive Search Procedure (GRASP) [40, 41] is used. In GRASP, each cycle consists of two stages: construction and local search. The construction stage builds a feasible solution whose neighborhood is investigated until a local minimum is found during the local search stage. Algorithm 2 shows the pseudo-code of the GRASP.

Algorithm 2: Initialization phase using GRASP

1. Input (Tasks, Virtual Machines).
2. Repeat for (all number of food sources (solutions))
 - 2.1. for Tasks $t = t_1, \dots, t_n$ do
 - 2.2. for machines $VMS = vm_1, \dots, vm_m$ do
 - 2.2.1. Choose randomly vm_k for t_j ;
 - 2.2.2. Solution \leftarrow Greedy Randomized Construction
 - 2.2.3. if Solution is not feasible then Solution \leftarrow Repair (Solution);
 - 2.3. end for.
 - 2.4. end for.
 - 2.5. Solution \leftarrow Local Search (Solution).
 - 2.6. Update Solution (Solution, Best Solution).
3. End.
4. Memorize the Best Solutions.

4.3 Allocation phase

The Binary Artificial Bee Colony (BABC) algorithm [12, 13] used in the proposed hybrid approach is designed based on the foraging behavior of honeybees (populations). It assumes number of food sources (or solutions) represent a trade-off among the objectives and works through optimizing them. The nectar amount of a food source corresponds to the quality (fitness) of the associated solution. There are three types of bees: employed, onlooker and scout bees. These food sources have been found by employee bees in the population (P_{uE}). Onlooker bees choose food sources probabilistically using the fitness (the multi-objective function) F_k in Eq. (15). The probability assigned to the K^{th} food source, P_k is calculated as Eq. (20):

$$P_k = \frac{F_k}{\sum_{i=1}^{P_{uE}} F_i} \quad (20)$$

When food sources (solutions) being found by employee bees could not get optimized in a predefined cycle (Limit), the employee bees will be abandoned and turn into the scout bees. A scout bee will search for a new food source (solution) using Algorithm 2 of GRASP. The main steps of BABC algorithm are given in Algorithm 3 and repeats until a predetermined termination criterion is met.

| Algorithm 3: Standard BABC algorithm for task scheduling |
|---|
| <ol style="list-style-type: none"> 1. Initialize. 2. Repeat. <ol style="list-style-type: none"> 2.1. Place the employed bees on the food sources in the search area. 2.2. Apply roulette wheel selection procedure to get fitness probability. 2.3. Place the onlooker bees on the food sources of higher probabilities. 2.4. Send the scouts to the search area for discovering new food sources. 3. UNTIL (requirements are met). |

4.4 Enhancing local search of the foragers

In the proposed hybrid approach, the employee bees in BABC algorithm use Right Circular Shift Neighborhood [42] for enhancing local search of the foragers. Right Circular Shift Neighborhood is obtaining by moving virtual machine assignments in the original food source (solution) one step in right direction then changing the virtual machine assignment of one task that got shifted out at one end and inserts it back at the other end. Further, the onlooker bees in BABC algorithm use Bit Inversion Mutation Operator [43] for maintaining sustainable diversity in a population. The mutation (swap) operator is applied in the food source (solution) among the tasks assigned to the virtual machine of the maximum completion time and the tasks assigned to the virtual machine that gives minimum completion time to generate the neighborhood solutions (food sources) of the bees. The food source (solution) is evaluated according

to fitness function in Eq. (15). Then the improvement rate (Im_R) is calculated as in Eq. (21):

$$Improvemet\ Rate(Im_R) = \frac{New(F) - Old(F)}{New(F)} 100\% \quad (21)$$

If Im_R is over 25% then the bee updates its food source (solution) to the new one and the trial count variable for the existing food source (solution) is setting to "0". Otherwise, if the bee does not change its food source, the trial count is incremented by 1. if Im_R is less than 25% and the trial count exceeds the limit then there is no improvement in the solution and this solution will be abandoned.

4.5 Producing pareto front based dominance approach

In the proposed approach, the solutions in the population are sorted after the termination criteria is met using a fast non-dominated sorting algorithm in [19]. Algorithm 4 presents the pseudo-code of the fast non-dominated sorting algorithm. The domination ranks ($Rank$) of the solution p (Sol_p) and the solution q (Sol_q) in the Bees population (P_B) are calculated based on two entities: The number of solutions which dominate the solution p (n_p), and the set of solutions that the solution p dominates (S_p).

| Algorithm 4: Fast non-dominated sorting algorithm. | |
|---|--|
| 1. | For each $Sl_p \in population (P_B)$ |
| 1.1. | $S_p = \emptyset$; |
| 1.2. | $n_p = 0$; |
| 1.3. | For each $(Sl_q) \in population (P_B)$ |
| 1.3.1. | If $Sl_p < Sl_q$ then |
| 1.3.1.1. | $S_p = S_p \cup \{q\}$; |
| 1.3.2. | else if $Sl_q < Sl_p$ then |
| 1.3.2.1. | $n_p = n_p + 1$; |
| 1.3.3. | End if; |
| 1.4. | If $n_p = 0$ then |
| 1.4.1. | $Rank(Sl_p) = 1$; |
| 1.4.2. | $F1 = F1 \cup \{Sl_p\}$; |
| 1.5. | End if; |
| 2. | $i = 1$; |
| 3. | while $F_i \neq \emptyset$ |
| 3.1. | $Q = \emptyset$ |
| 3.2. | For each $(Sl_p) \in F_i$ |
| 3.2.1. | For each $(Sl_q) \in S_p$ |
| 3.2.1.1. | $n_q = n_q - 1$; |
| 3.2.1.2. | If $n_q = 0$ then |
| 3.2.1.2.1. | $Rank(Sl_q) = i + 1$; |
| 3.2.1.2.2. | $Q = Q \cup \{Sl_q\}$; |
| 3.3. | $i = i + 1$; |
| 3.4. | $F_i = Q$; |

4.6 The proposed algorithm EBABC-PF

Algorithm 5 presents the pseudo-code of the proposed hybrid algorithm.

Algorithm 5: The proposed hybrid algorithm (EBABC-PF).

1. Build task priority list using the **HEFT** in **Algorithm 1**.
2. Initialize food source Population (feasible solutions) using the **GRASP** in **Algorithm 2**.
3. **REPEAT**
 - 3.1. Place the employed bees on the food sources in the search area.
 - 3.2. **For each employee bee** in the Population
 - 3.2.1. Evaluate the current solution using Equations (11-18).
 - 3.2.2. Produce neighborhood solution using **Right Circular Shift** as described in subsection 4-4.
 - 3.2.3. Calculate the improvement rate for each employee bee using Equation (21).
 - 3.2.4. Abandon the solution if the trial count exceeds the limit or the improvement is less than 25%.
 - 3.2.5. Calculate the probability function for the solutions in search area using Equation (20).
 - 3.2.6. Memorize the best solutions.
 - 3.3. Place the onlooker bees on the food sources in the search area.
 - 3.4. **For each onlooker bee** in the Population
 - 3.4.1. Evaluate the current solution using Equations (11-18).
 - 3.4.2. Produce neighborhood solution using **Bit Mutation Operators** as described in subsection 4-4.
 - 3.4.3. Calculate the improvement rate for each onlooker bee using Equation (21).
 - 3.4.4. Abandon the solution if the trial count exceeds the limit or the improvement is less than 25%.
 - 3.5. Send scout bees to the search area.
 - 3.6. Replace the abandoned solutions using **GRASP** in **Algorithm 2**.
 - 3.7. Apply **fast non-dominated sorting algorithm** to the population as in **Algorithm 4**.
 - 3.8. Evaluate the current solution using Equation (11-18).
 - 3.9. Select the best non-dominated solutions.
4. **UNTIL** (requirements are met).

4.7 The complexity analysis of the proposed hybrid algorithm

This section illustrates the time complexity of the proposed hybrid algorithm. According to Algorithm 5, the time complexity is the summation of the time complexities of Algorithms 1, 2, 3 and 4. Let n be the number of tasks, m is the number of nodes in the clouds, Pu_E is the number of the food sources (feasible solutions) in the population, and φ is the number of objectives to be optimized. In Algorithm 1, task priority list is built by HEFT algorithm hence, the time complexity is $O(n)$. In Algorithm 2, the feasible solutions in the population are initialized using the GRASP therefore this algorithm require $sO(Pu_E mn)$. In Algorithm 3, the allocation phase uses BABC algorithm modified with Right Circular Shift method at employee bees phase while Bit Mutation Operators is used in onlooker bees phase. Third, send the scouts search the area using the GRASP. This allocation step requires time complexity $O(2Pu_E mn^2 + Pu_E mn) \cong O(Pu_E mn^2)$. Finally, apply fast non-dominated sorting algorithm requires $O(\varphi Pu_E^2)$.

In our experiment, the overall complexity of the proposed hybrid algorithm is $O(n + Pu_E mn + Pu_E mn^2 + \varphi Pu_E^2) \cong O(Pu_E mn^2 + Pu_E^2)$ assuming the three objectives ($\varphi = 3$): makespan (MS), cost (TC^{exe}) and utilization ($A\hat{u}$). During our experiment, we assume that the number of the food sources in the population $Pu_E = 40$ and it is fixed then the overall complexity is simplified to be $O(Pu_E mn^2 + Pu_E^2) \cong O(mn^2)$.

5 Experiments

This section describes the overall experimental setup, performance metrics, results, and analysis to evaluate the proposed hybrid approach (EBABC-PF). We compared the performance of the proposed approach with Heterogeneous Earliest Finish Time (HEFT) [10, 11], Deadline Heterogeneous Earliest Finish Time (DHEFT) algorithm [44], Fast and Elitist Multi-objective Genetic Algorithm (NSGA-II) [19, 43] and Binary Artificial Bee Colony (BABC) algorithm [12, 13].

5.1 Performance metrics

Based on our simulation setup, we have four performance metrics computed by formulas (12–15):

1. *The makespan (MS)*: The overall completion time needed to execute all the tasks by the available clouds.
2. *The resource utilization ($A\hat{u}$)*: The ratio of the average completion time of the virtual machine and the overall makespan.
3. *The processing cost (TC^{exe})*: The total processing cost needed to execute submitted application using the available clouds.

5.2 Environment setup

The experiments were carried out using NetBeans IDE Version 8.0.2, CloudSim version-3.0 and WorkflowSim Version-1.1.0. Experimental environment includes Intel(R)-Core(TM)i7-7500U-2.70 GHz processor and 16.0 GB RAM running on Microsoft Windows 10. Our proposed simulation carried out the scheduling process coded in Java and evaluated through simulation run for Montage, CyberShake, EpiGenomics, LIGO Inspiral, and SIPHT workflows, respectively.

5.3 Parameters settings

In these experiments, we have determined the population parameters and various conditions according to the implemented experiment which influence the proposed algorithm EBABC-PF. Using Amazon EC2 instance pricing list and Amazon charges proposed in [45] for on-demand reserved virtual machines instances on an hourly basis, we calculate the cost accordingly. The suggested parameter settings for the proposed approach EBABC-PF are provided in Tables [1–3]. We assume the cloud data centers setting with the host specifications which are presented in Tables 1 and 2, respectively. The suggested configuration for the proposed approach EBABC-PF is provided in Table 3.

Table 1 Data center configuration

| Parameter name | Parameter values |
|----------------------------------|------------------|
| Number of data center | 3 |
| Number of virtual machines | {20, 80} |
| Operating system (OS) | LINUX |
| Virtual machine hypervisor (VMM) | Xen |
| Cost per vm(\$/Hr) | 2.224 |

Table 2 Simulated host specification

| Host type | CPU(MIPS) | Storage-RAM-BW |
|---------------------|---------------------|--|
| A dual-core machine | 1000–1500 | Storage(1 GB) RAM(2048 MB) BW(103 b/s) |
| A quad-core machine | 1000–1500-2000–2500 | Storage(1 GB) RAM(2096 MB) BW(104 b/s) |

Table 3 Final configuration of the proposed algorithm

| | |
|-----------------------|---|
| Population size | 40 |
| Employee bee | 20 |
| Onlooker bees | 20 |
| Scout bees | The best solutions replace the worst ones |
| The maximum iteration | 100 |
| The Limit | 5 |

5.4 Experimental results with different scientific workflows

The workflows used in our experiment are synthesized based on the benchmark workflows available in the WorkflowSim simulation tool. Five groups of practical workflows benchmarks are chosen, i.e., Montage, CyberShake, Epigenomics, LIGO, and SIPHT [36, 37]. In our experiment, two different sizes of these workflows around the tasks number in [100, 1000] are utilized while the available virtual machines number are in [20, 80].

To measure the effectiveness of our proposed algorithm EBABC-PF, the performance of the proposed approach is compared with Heterogeneous Earliest Finish Time (HEFT) [10, 11] algorithm, Deadline Heterogeneous Earliest Finish Time (DHEFT)

[44] algorithm, Non-dominated Sort Genetic Algorithm (NSGA-II) [19, 43] as we have taken binary tournament selection, one-point crossover with the mutation process, and Standard Binary Artificial Bee Colony (BABC) [12, 13] algorithm.

These algorithms are implemented in WorkflowSim simulator using the benchmark workflows. Tables 4, 5 and 6 show the experimental results with different scientific workflows in terms of makespan (MS), resource utilization ($A\hat{u}$) and processing cost (TC^{exe}). A comparison of the makespan (MS) generated among the algorithms: HEFT, DHEFT, NSGA-II, BABC, and EBABC-PF using the benchmark workflows is shown in Table 4. The resource utilization ($A\hat{u}$) comparison generated among the algorithms: HEFT, DHEFT, NSGA-II, BABC, and EBABC-PF is shown in Table 5. The processing cost (TC^{exe}) comparison among the algorithms: HEFT, DHEFT, NSGA-II, BABC, and the proposed algorithm EBABC-PF is shown in Table 6 using the benchmark workflows. It is obvious that the proposed hybrid algorithm EBABC-PF gives the best results in terms of makespan (MS), utilization ($A\hat{u}$) and cost (TC^{exe}) compared to HEFT, DHEFT, NSGA-II, and BABC for all instances.

Table 4 The minimum makespan(sec.) obtained by the number of virtual machines with HEFT, DHEFT, NSGA-II, BABC and the proposed approach EBABC-PF

| Applications | Tasks No | Virtual machines No | HEFT | DHEFT | NSGA-II | BABC | EBABC-PF |
|------------------|----------|---------------------|---------|---------|---------|---------|----------|
| Montage | 100 | 20 | 256.500 | 214.500 | 182.570 | 150.278 | 106.009 |
| | 100 | 80 | 117.150 | 97.968 | 83.384 | 68.636 | 48.417 |
| | 1000 | 20 | 989.568 | 863.200 | 762.345 | 632.100 | 521.260 |
| | 1000 | 80 | 896.390 | 781.921 | 690.562 | 572.581 | 472.178 |
| CyberShake | 100 | 20 | 326.700 | 276.400 | 216.980 | 176.100 | 136.400 |
| | 100 | 80 | 227.560 | 192.524 | 151.136 | 122.661 | 95.008 |
| | 1000 | 20 | 665.452 | 373.567 | 303.342 | 253.762 | 215.672 |
| | 1000 | 80 | 497.869 | 279.490 | 226.950 | 189.856 | 161.359 |
| Epigenomics | 100 | 20 | 465.023 | 378.428 | 303.340 | 253.764 | 198.852 |
| | 100 | 80 | 196.552 | 159.951 | 128.213 | 107.259 | 84.049 |
| | 1000 | 20 | 856.540 | 632.789 | 458.972 | 352.546 | 252.276 |
| | 1000 | 80 | 479.489 | 354.234 | 256.931 | 197.354 | 141.223 |
| LIGO Inspiral | 100 | 20 | 389.879 | 318.907 | 289.190 | 209.880 | 196.280 |
| | 100 | 80 | 174.964 | 143.114 | 129.778 | 94.187 | 88.084 |
| | 1000 | 20 | 689.568 | 563.345 | 492.742 | 402.198 | 372.260 |
| | 1000 | 80 | 397.667 | 324.875 | 284.159 | 231.944 | 214.679 |
| SIPHT | 100 | 20 | 296.860 | 183.410 | 104.320 | 86.472 | 65.280 |
| | 100 | 80 | 141.387 | 87.354 | 49.685 | 41.184 | 31.091 |
| | 1000 | 20 | 529.568 | 453.345 | 362.345 | 265.100 | 210.289 |
| | 1000 | 80 | 290.385 | 248.589 | 198.689 | 145.366 | 115.311 |

Table 5 The utilization (%) obtained by the number of virtual machines with HEFT, DHEFT, NSGA-II, BABC and the proposed approach EBABC-PF

| Applications | Tasks No | Virtual machines No | HEFT | DHEFT | NSGA-II | BABC | EBABC-PF |
|------------------|----------|---------------------|-------|-------|---------|-------|----------|
| Montage | 100 | 20 | 49.67 | 52.89 | 58.89 | 67.56 | 78.76 |
| | 100 | 80 | 38.46 | 40.95 | 45.60 | 52.31 | 60.99 |
| | 1000 | 20 | 61.67 | 67.89 | 72.89 | 78.56 | 81.76 |
| | 1000 | 80 | 54.62 | 60.12 | 64.55 | 69.57 | 72.41 |
| CyberShake | 100 | 20 | 53.89 | 60.26 | 67.34 | 72.67 | 85.45 |
| | 100 | 80 | 49.21 | 55.03 | 61.49 | 66.36 | 78.03 |
| | 1000 | 20 | 70.89 | 76.26 | 80.34 | 83.67 | 89.45 |
| | 1000 | 80 | 68.98 | 74.21 | 78.18 | 81.42 | 87.04 |
| Epigenomics | 100 | 20 | 52.67 | 61.85 | 69.67 | 78.34 | 87.45 |
| | 100 | 80 | 48.34 | 56.77 | 63.94 | 71.90 | 80.26 |
| | 1000 | 20 | 69.67 | 73.67 | 80.67 | 87.34 | 93.45 |
| | 1000 | 80 | 67.41 | 71.27 | 78.05 | 84.50 | 90.41 |
| LIGO Inspiral | 100 | 20 | 38.89 | 48.26 | 52.34 | 59.67 | 62.45 |
| | 100 | 80 | 22.69 | 28.16 | 30.54 | 34.81 | 36.44 |
| | 1000 | 20 | 48.89 | 54.26 | 59.34 | 65.67 | 69.45 |
| | 1000 | 80 | 32.82 | 36.42 | 39.84 | 44.08 | 46.62 |

5.5 The performance evaluation using benchmark workflows

In our experiments, we consider three conflicting objectives, minimizing makespan and processing cost along with maximizing the resource utilization. Our proposed algorithm EBABC-PF schedules the workflows onto the available virtual machines taking into account the load balance among the available virtual machines by swapping the tasks allocated to the virtual machine with the minimum completion time and the tasks allocated to the virtual machine with the maximum completion time within the search process. After predefined cycles for running search process in our experiment and based on the dominance approach, the final population (feasible solutions) is sorted. The best near-optimal solution is selected from the final sorted population according to the value of the multi-objective function F_k formulated as Eq. (15).

A statistical analysis of the outputs when scheduling the benchmark workflows (Montage, CyberShake, Epigenomics, LIGO, and SIPHT) applications is shown in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 considering different sizes of these workflows around the interval [100,1000] tasks and the available virtual machines are in the interval [20,80]. The makespan results of the benchmark workflows used in our experiment are depicted in Figs. 4, 5, 6, 7 and 8. When applying our proposed algorithm EBABC-PF, the decreasing rate in makespan(MS) ranges from -78.01% to -6.48% . The resource utilization results of the benchmark workflows are shown in

Table 6 The processing cost (\$) obtained by HEFT, DHEFT, NSGA-II, BABC and the proposed approach (processing cost per vm = 2.224\$)

| Applications | Tasks No | Virtual machines No | HEFT | DHEFT | NSGA-II | BABC | EBABC-PF |
|---------------|----------|---------------------|---------|---------|---------|---------|----------|
| Montage | 100 | 20 | 1809.36 | 1273.19 | 953.62 | 734.23 | 653.11 |
| | 100 | 80 | 4227.22 | 2974.56 | 2227.95 | 1715.39 | 1525.87 |
| | 1000 | 20 | 2682.75 | 2484.07 | 2093.07 | 1868.77 | 1649.30 |
| | 1000 | 80 | 5359.06 | 4962.18 | 4181.12 | 3733.06 | 3294.64 |
| CyberShake | 100 | 20 | 2274.31 | 1825.67 | 1507.78 | 1247.82 | 959.89 |
| | 100 | 80 | 4726.11 | 3793.82 | 3133.23 | 2593.02 | 1994.69 |
| | 1000 | 20 | 3588.99 | 2996.98 | 2758.04 | 2443.63 | 2205.16 |
| | 1000 | 80 | 5610.87 | 4685.35 | 4311.79 | 3820.26 | 3447.45 |
| Epigenomics | 100 | 20 | 2851.31 | 2315.67 | 2123.78 | 1847.82 | 1459.89 |
| | 100 | 80 | 5136.07 | 4171.22 | 3825.57 | 3328.48 | 2629.70 |
| | 1000 | 20 | 4484.09 | 3928.46 | 3372.90 | 2867.33 | 2539.53 |
| | 1000 | 80 | 5831.06 | 5108.53 | 4386.09 | 3728.65 | 3302.38 |
| LIGO Inspiral | 100 | 20 | 2176.78 | 1850.59 | 1229.37 | 989.72 | 686.88 |
| | 100 | 80 | 4763.76 | 4049.91 | 2690.41 | 2165.95 | 1503.20 |
| | 1000 | 20 | 3685.81 | 3274.63 | 2887.37 | 2260.72 | 1971.81 |
| | 1000 | 80 | 5163.76 | 4587.70 | 4045.16 | 3167.23 | 2762.47 |
| SIPHT | 100 | 20 | 1751.78 | 1423.63 | 1298.21 | 979.47 | 657.44 |
| | 100 | 80 | 4284.46 | 3481.88 | 3175.13 | 2395.56 | 1607.95 |
| | 1000 | 20 | 3171.72 | 2871.66 | 2570.74 | 2116.08 | 1790.49 |
| | 1000 | 80 | 4807.16 | 4352.37 | 3896.29 | 3207.19 | 2713.72 |

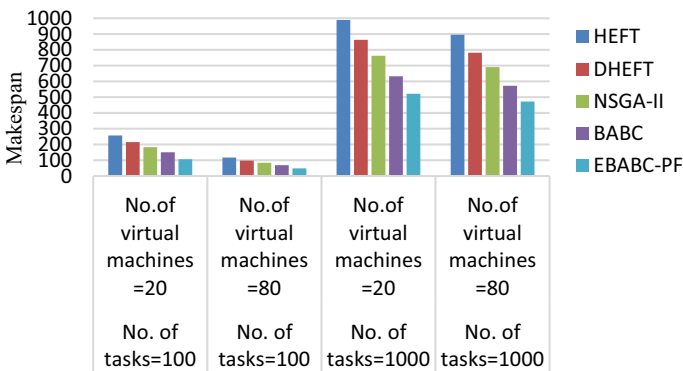


Fig. 4 The minimum makespan(sec.) for Montage workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

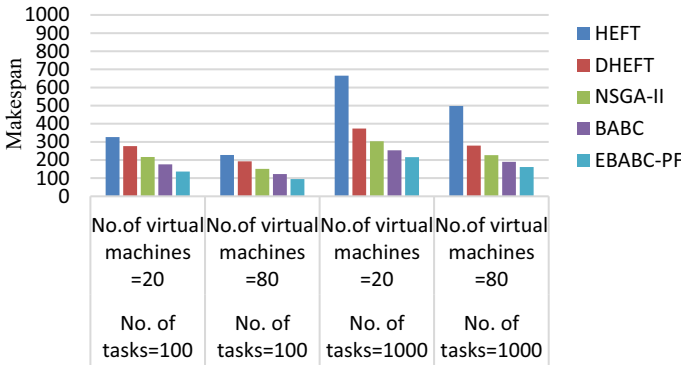


Fig. 5 The minimum makespan(sec.) for CyberShake workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

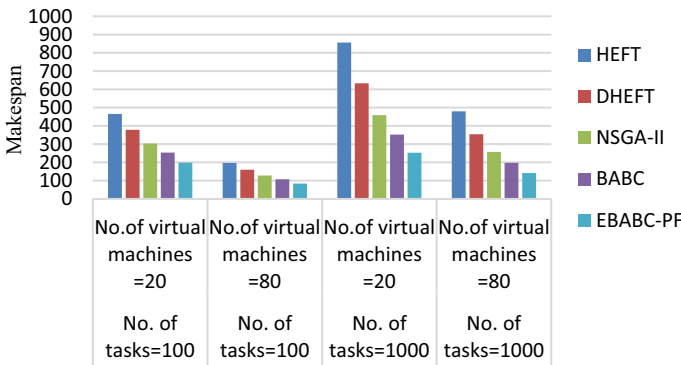


Fig. 6 The minimum makespan(sec.) for Epigenomics workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

Figs. 9, 10, 11, 12 and 13. The increasing rate in utilization ($A\hat{u}$) ranges from 4.07% to 66.03%. We plot monetary cost in Figs. 14, 15, 16, 17 and 18. The processing cost (TC^{exe}) decreasing rate ranges from -68.45% to -9.76%.

For example, comparable with BABC algorithm when scheduling the benchmark workflows using our proposed algorithm EBABC-PF, there are different decreasing rates in makespan(MS) according to the different number of the tasks summited. When scheduling Montage workflow using EBABC-PF, the decreasing rate in makespan(MS) is -29.46%. Although when the number of tasks equal to 1000, the decreasing rate in makespan(MS) is -17.54%. If scheduling CyberShake workflow using EBABC-PF comparable with BABC algorithm, the decreasing rate in makespan(MS) is -22.54% with number of tasks equal to 100, while it is -15.01% with number of tasks equal to 1000. Scheduling Epigenomics workflow using EBABC-PF achieves decreasing rate in makespan(MS) about -21.64% comparable with

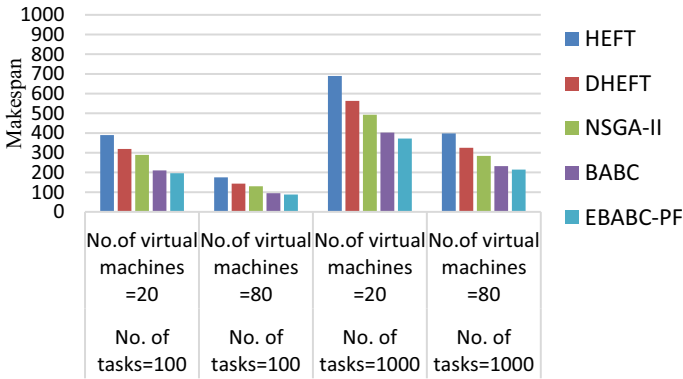


Fig. 7 The minimum makespan(sec.) for LIGO Inspiral workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

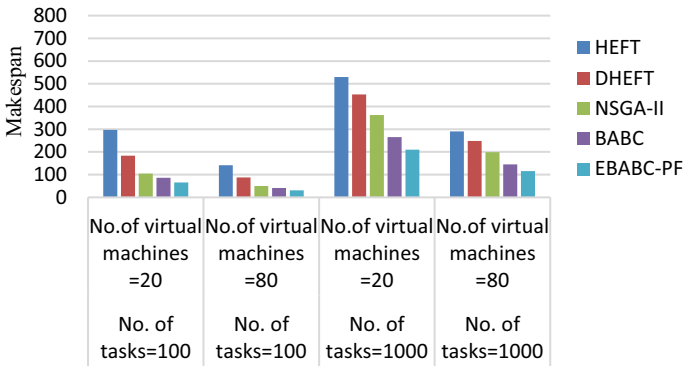


Fig. 8 The minimum makespan(sec.) for SIPHT workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

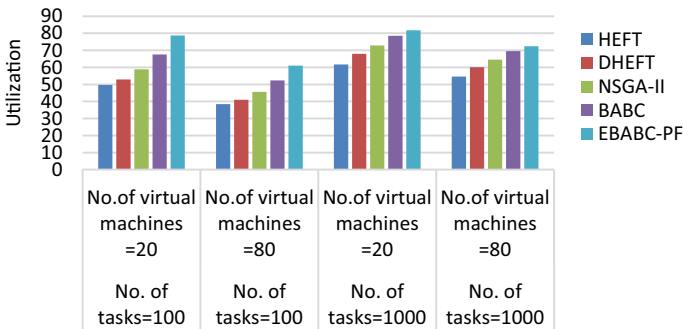


Fig. 9 The average resource utilization(%) for Montage workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

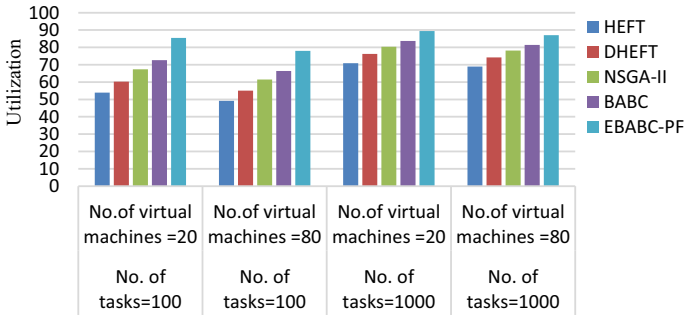


Fig. 10 The average resource utilization(%) for CyberShake workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

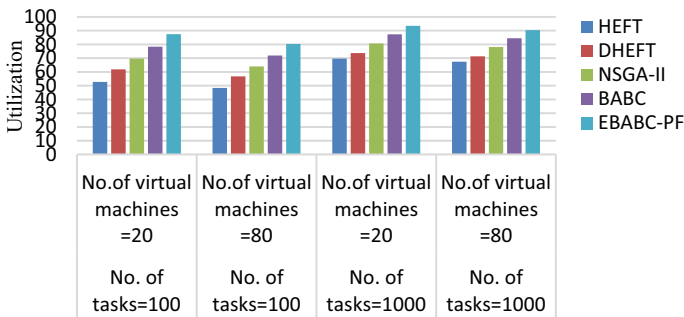


Fig. 11 The average resource utilization(%) for Epigenomics workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

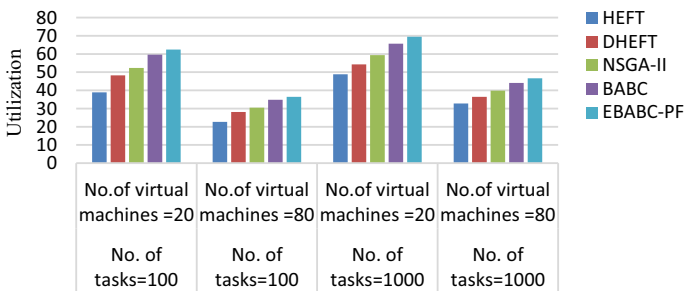


Fig. 12 The average resource utilization(%) for LIGO Inspiral workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

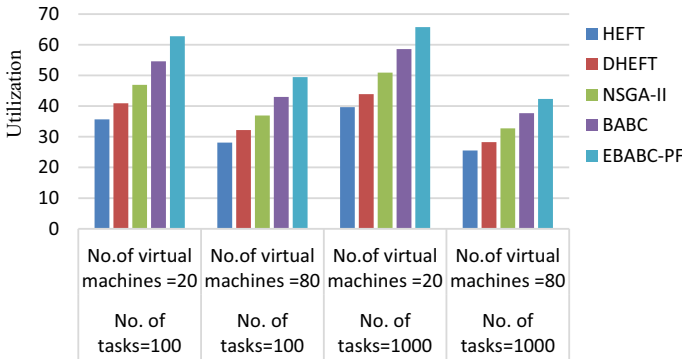


Fig. 13 The average resource utilization(%) for SIPHT workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}

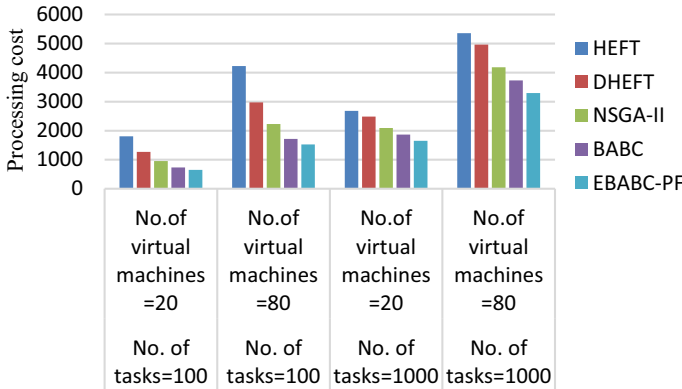


Fig. 14 Processing cost for Montage workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100, 1000} and number of virtual machines = {20, 80}. (processing cost per vm = 2.224\$/hr)

BABC. The decreasing rate is -28.44% with number of tasks equal to 1000. Scheduling LIGO workflow using EBABC-PF algorithm has decreasing rate in makespan(MS) is about -6.48% comparable with BABC, while it is -7.44% if number of tasks is 1000. When scheduling SIPHT workflow using EBABC-PF algorithm with number of tasks equal to 100, the decreasing rate in makespan(MS) is -24.51% comparable with BABC algorithm while the decreasing rate is -20.68% with number of tasks equal to 1000.

Overall, comparable with BABC algorithm when scheduling the benchmark workflows (Montage, CyberShake, Epigenomics, LIGO, and SIPHT), the decreasing rate in makespan(MS) of EBABC-PF algorithm is decreasing with the number of tasks is increasing in the case of Montage, CyberShake, and SIPHT applications while the

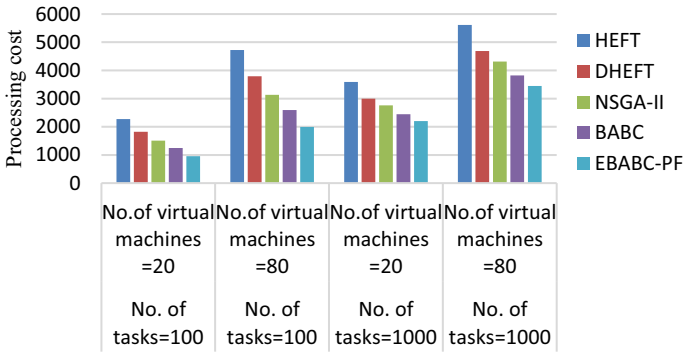


Fig. 15 Processing cost for CyberShake workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = { 100,1000} and number of virtual machines = {20,80} . (processing cost per vm = 2.224\$/hr)

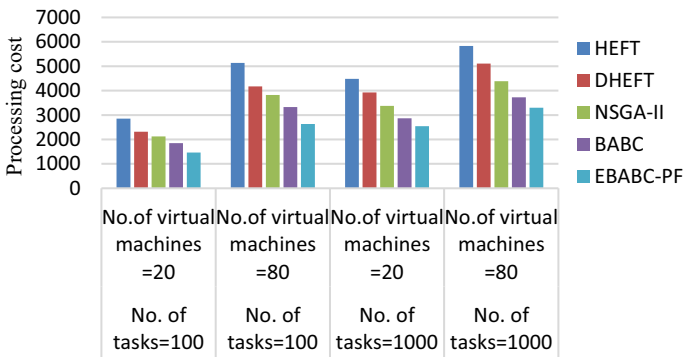


Fig. 16 Processing cost for Epigenomics workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = { 100,1000} and number of virtual machines = {20,80} . (processing cost per vm = 2.224\$/hr)

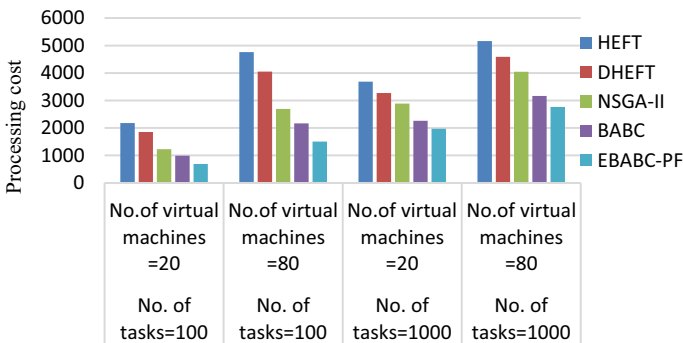


Fig. 17 Processing cost for LIGO Inspiral workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = { 100, 1000} and number of virtual machines = {20, 80} . (processing cost per vm = 2.224\$/hr)

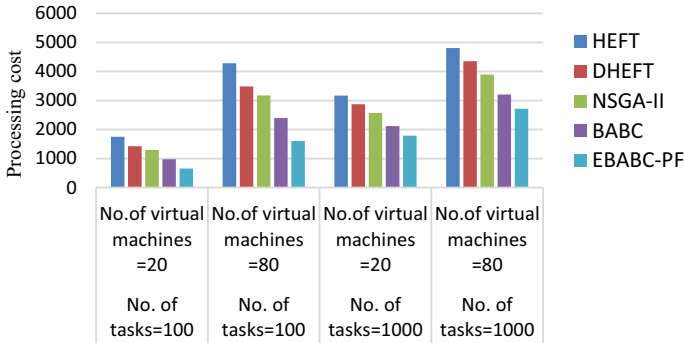


Fig. 18 Processing cost for SIPHT workflow application obtained by HEFT, DHEFT, NSGA-II, BABC and EBABC-PF with number of tasks = {100,1000} and number of virtual machines = {20,80}. (processing cost per vm = 2.224\$/hr)

decreasing rate is increasing with the number of tasks increase in the case of Epigenomics workflow and LIGO workflow.

Likewise, there are different increasing rates in utilization ($A\hat{u}$) according to the different number of the tasks summited when scheduling the benchmark workflows (Montage, CyberShake, Epigenomics, LIGO, and SIPHT) applications using our proposed algorithm EBABC-PF comparable with BABC algorithm. When scheduling Montage workflow using our proposed algorithm EBABC-PF, the increasing rate in utilization ($A\hat{u}$) is 16.58% comparable with BABC algorithm if the number of tasks is equal to 100 while the increasing rate is 4.07% if the number of tasks is 1000. When scheduling CyberShake workflow with EBABC-PF, the increasing rate in utilization ($A\hat{u}$) is 17.59% comparable with BABC algorithm if the number of tasks is 100 while the increasing rate is 6.91% with the number of tasks equal to 1000. If scheduling Epigenomics workflow using EBABC-PF, it achieves increasing rate in utilization ($A\hat{u}$) about 11.63% while the increasing rate is 7.00% with number of tasks equal to 1000 comparable with BABC algorithm. Scheduling LIGO workflow using EBABC-PF achieves increasing rate in utilization ($A\hat{u}$) about 4.66% comparable with BABC if the number of tasks is 100, while it is 5.76% with the number of tasks equal to 1000. When scheduling SIPHT workflow using EBABC-PF algorithm with number of tasks equal to 100, the increasing rate in utilization ($A\hat{u}$) comparable with BABC algorithm is 4.66% while it is 5.76% with the number of tasks equal to 1000.

We note that when scheduling such workflows, the increasing rate in utilization ($A\hat{u}$) of our proposed algorithm EBABC-PF is decreasing with the number of tasks increase in the case of Montage, CyberShake, and Epigenomics while the increasing rate of utilization ($A\hat{u}$) is increasing with the number of tasks increase in the case of SIPHT workflow and LIGO workflow.

Furthermore, there are different decreasing rates in the processing cost (TC^{exe}) according to the different number of the tasks summited when scheduling the

benchmark workflows (Montage, CyberShake, Epigenomics, LIGO, and SIPHT) applications using our proposed algorithm EBABC-PF comparable with BABC algorithm. When scheduling Montage workflow with number of tasks equal to 100 using EBABC-PF algorithm, the decreasing rate in the processing cost (TC^{exe}) is -11.05% while the decreasing rate in the processing cost (TC^{exe}) is -11.74% when the number of tasks equal to 1000. When scheduling CyberShake workflow using EBABC-PF algorithm, the decreasing rate in cost (TC^{exe}) is -23.07% with the number of tasks equal to 100 while the decreasing rate is -9.76% with the number of tasks equal to 1000. If scheduling Epigenomics workflow, the decreasing rate is about -20.99% with the number of tasks equal to 100 while the decreasing rate is -11.43% with the number of tasks equal to 1000. While scheduling LIGO workflow using EBABC-PF algorithm, the decreasing rate in the processing cost (TC^{exe}) is -30.60% while it is -12.78% with the number of tasks equal to 1000 comparable with BABC algorithm. When scheduling SIPHT workflow the decreasing rate in the processing cost (TC^{exe}) is -32.88% with the number of tasks equal to 100, while the decreasing rate is -15.39% with the number of tasks equal to 1000.

Generally, with this workflow scheduling using our proposed algorithm EBABC-PF, the decreasing rate in the processing cost (TC^{exe}) comparable to BABC algorithm is decreasing with the number of tasks increase in the case of CyberShake, Epigenomics, LIGO and SIPHT workflow applications while the decreasing rate is increasing slightly with the number of tasks increase in the case of Montage workflow application only.

Practically, the results depict that our proposed algorithm EBABC-PF outperforms all the suggested baseline algorithms for comparison (HEFT, DHEFT, NSGA-II, and BABC) considering minimizing makespan (MS) and processing cost (TC^{exe}) while maximizing the utilization ($A\hat{u}$). Although, BABC shows better results than HEFT, DHEFT, and NSGA-II. While NSGA-II is better than DHEFT but DHEFT outperforms HEFT. Overall, our proposed algorithm EBABC-PF gives noticeable improvement for every type of the benchmark workflows implemented in our experiments.

6 Conclusion and future work

Scheduling workflow applications in cloud computing is an important issue as a multi-objective optimization problem that need an efficient scheduling strategy to optimize the use of cloud resources considering different Quality of Services (QoS) requirements. This article develops a new efficient hybrid algorithm for multi-objective workflow scheduling problem in cloud, called EBABC-PF. In our approach, the priority list is built for the summited workflow applications using the Heterogeneous Earliest Finish Time (HEFT) algorithm. As the determination of the initial population plays a crucial role to preserve convergence and diversity in the search space of the food sources (feasible solutions), the proposed approach combines the greedy randomized adaptive search procedure (GRASP) and the Binary Artificial Bee Colony (BABC). Further, several modifications are applied considering the loads of the available virtual machines within swapping tasks between the virtual machine of maximum completion

time and the other of minimum completion time in each food source (feasible solution). These modifications try to enhance local search of the foragers and maintaining sustainable diversity in a population towards minimizing makespan, and processing cost while maximizing utilization. The proposed approach is simulated using WorkflowSim simulator based CloudSim simulator and the results are compared with the algorithms: HEFT, DHEFT, NSGA-II, BABC. The simulation results and the comparisons demonstrate the effectiveness of the proposed algorithm EBABC-PF in terms of makespan, processing cost and utilization compared with the other existing algorithms.

The future work is to investigate other meta-heuristics for the multi-objective task scheduling problems and generalize the application of the proposed algorithm to solve other combinatorial optimization problems considering the evolution of power-constrained Internet of Things (IoT) devices.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflict of interest All Authors, Eng. Maha Zeedan, Prof. Gamal Attiya, and Prof. Nawal El-Fishawy declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Buyya R, James B, Andrzej MG (eds) (2010) Cloud computing: principles and paradigms, vol 87. Wiley, New York
2. Rodriguez-Maria A, Rajkumar B (2017) A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurr Comput Pract Exp* 29(8):e4041
3. Bernstein D, Vij D and Diamond S (2011) An intercloud cloud computing economy-technology, governance, and market blueprints. In: Annual SRII global conference. IEEE
4. Singh P, Dutta M, Aggarwal N (2017) A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowl Inf Syst* 52(1):1–51
5. El-Ghazali T (2009) Metaheuristics: from design to implementation, vol 74. Wiley, New York
6. Resende MGC, Ribeiro CC (1998) Greedy randomized adaptive search procedures (GRASP). *AT&T Labs Res Tech Rep* 98(1):1–11
7. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Nat Comput* 17(3):585–609
8. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. In: Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Vol 200

9. Baykasoglu A, Ozbakir L, and Tapkan P (2007) Artificial bee colony algorithm and its application to generalized assignment problem. In: *Swarm intelligence: focus on ant and particle swarm optimization 1*
10. Samadi Y, Zbakh M, and Tadonki C (2018) E-HEFT: enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing. In: *International conference on high performance computing & simulation (HPCS)*. IEEE
11. Mazrekaj A et al. (2019) The experiential heterogeneous earliest finish time algorithm for task scheduling in clouds. In: *CLOSER*
12. Baykasoglu A, Özbakır L, Tapkan P (2007) Artificial bee colony algorithm and its application to generalized assignment problem. *Swarm Intell Focus Ant Part Swarm Optim* 1:1–10
13. Li J-Q, Han Y-Q (2019) A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust Comput* 23:1–17
14. Calheiros RN et al (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp* 41(1):23–50
15. Chen W and Deelman E (2012) WorkflowSim: a toolkit for simulating scientific workflows in distributed environments. In: *IEEE 8th international conference on E-science*. IEEE
16. Zuo L et al (2015) A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access* 3:2687–2699
17. Kimpan W and Kruekaew B (2016) Heuristic task scheduling with artificial bee colony algorithm for virtual machines. In: *Joint 8th international conference on soft computing and intelligent systems (SCIS) and 17th international symposium on advanced intelligent systems (ISIS)*. IEEE
18. Wu Q et al (2016) Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. *IEEE Trans Serv Comput* 12:550–563
19. Deb K et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 6(2):182–197
20. Sofia AS, Kumar PG (2018) Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. *J Netw Syst Manag* 26(2):463–485
21. Naik K, Gandhi GM, Patil SH (2019) "Multiobjective virtual machine selection for task scheduling in cloud computing. *Computational intelligence: theories applications and future directions-volume I*. Springer, Singapore, pp 319–331
22. Gao Y, Zhang S, Zhou J (2019) A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud. *IEEE Access* 7:125783–125795
23. Marahatta A et al (2019) Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center. *IEEE Trans Cloud Comput* 9:1376–1390
24. Shen L et al. (2019) Optimization of artificial bee colony algorithm based load balancing in smart grid cloud. In: *IEEE innovative smart grid technologies-asia (ISGT Asia)*, IEEE
25. Kruekaew B, Kimpan W (2020) Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. *Int J Comput Intell Syst* 13(1):496–510
26. Farid M et al (2020) Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. *IEEE Access* 8:24309–24322
27. Wang J et al (2020) Energy utilization task scheduling for mapreduce in heterogeneous clusters. *IEEE Trans Serv Comput*. <https://doi.org/10.1109/TSC.2020.2966697>
28. Yuan H et al (2020) Biobjective task scheduling for distributed green data centers. *IEEE Trans Autom Sci Eng* 18:731–742
29. Verma A, Kaushal S (2017) A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Comput* 62:1–19
30. Ijaz S et al (2021) Energy-makespan optimization of workflow scheduling in fog–cloud computing. *Computing* 103:1–27
31. Kaur G, Bala A (2021) Prediction based task scheduling approach for floodplain application in cloud environment. *Computing* 103:1–22
32. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J Glob Optim* 39(3):459–471
33. Pham DT, Castellani M (2015) A comparative study of the bees algorithm as a tool for function optimisation. *Cogent Eng* 2(1):1091540
34. Kim S-S et al (2017) Cognitively inspired artificial bee colony clustering for cognitive wireless sensor networks. *Cogn Comput* 9(2):207–224

35. Pandey S, Karunamoorthy D, Buyya R (2011) Workflow engine for clouds. *Cloud Comput Princ Paradig* 87:321–344
36. Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2012) Characterizing and profiling scientific workflows. *Future Gener Comput Syst* 29(3):682–692
37. Yu J, Buyya R, Ramamohanarao K (2008) Workflow scheduling algorithms for grid computing. *Metaheuristics for scheduling in distributed computing environments*. Springer, Berlin, Heidelberg, pp 173–214
38. Kumar M et al (2019) A comprehensive survey for scheduling techniques in cloud computing. *J Netw Comput Appl* 143:1–33
39. Buyya R, Ranjan R, and Calheiros RN (2009) Modeling and simulation of scalable Cloud computing environments and the cloudsim toolkit: challenges and opportunities. In: *International conference on high performance computing & simulation*. IEEE
40. Shao W, Pi D, and Shao Z (2017) A hybrid iterated greedy algorithm for the distributed no-wait flow shop scheduling problem. In: *IEEE congress on evolutionary computation (CEC)*. IEEE
41. Resende MGC, Ribeiro CC (2019) "Greedy randomized adaptive search procedures: advances and extensions." *Handbook of metaheuristics*. Springer, Cham, pp 169–220
42. Bird J (2017) *Basic engineering mathematics*. Routledge, New York
43. Poli R et al (2008) *A field guide to genetic programming*. Springer, Cham
44. Rahman M, Ranjan R, and Buyya R (2009) A distributed heuristic for decentralized workflow 646 scheduling in global grids. In: *IEEE/ACM international conference on grid computing*. IEEE
45. <https://aws.amazon.com/ec2/pricing/on-demand/>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.