The Journal of Supercomputing (2022) 78:17423–17449
https://doi.org/10.1007/s11227-022-04539-8

GSAGA: A hybrid algorithm for task scheduling in cloud infrastructure

Poria Pirozmand[1] · Amir Javadpour[2,3] · Hamideh Nazarian[4] · Pedro Pinto[2] · Seyedsaeid Mirkamali[5] · Forough Ja'fari[6]

Accepted: 13 April 2022 / Published online: 19 May 2022

Abstract

Cloud computing is becoming a very popular form of distributed computing, in which digital resources are shared via the Internet. The user is provided with an overview of many available resources. Cloud providers want to get the most out of their resources, and users are inclined to pay less for better performance. Task scheduling is one of the most important aspects of cloud computing. In order to achieve high performance from cloud computing systems, tasks need to be scheduled for processing by appropriate computing resources. The large search space of this issue makes it an NP-hard problem, and more random search methods are required to solve this problem. Multiple solutions have been proposed with several algorithms to solve this problem until now. This paper presents a hybrid algorithm called GSAGA to solve the Task Scheduling Problem (TSP) in cloud computing. Although it has a high ability to search the problem space, the Genetic Algorithm (GA) performs poorly in terms of stability and local search. It is therefore possible to create a stable algorithm by combining the general search capacities of the GA with the Gravitational Search Algorithm (GSA). Our experimental results indicate that the proposed algorithm can solve the problem with higher efficiency compared with the state-of-the-art.

Keywords Cloud computing · Scheduling · Resources · SLA · Epigenomics · GSA · NP-hard problem · Genetic algorithm · Gravitational search

✉ Amir Javadpour
a.javadpour87@gmail.com; a_javadpour@e.gzhu.edu.cn

Extended author information available on the last page of the article

1 Introduction

Research has become increasingly interested in using cloud computing for various applications, including telecommunications, manufacturing, education, and academic research in recent years [1, 2]. A storage cloud, for instance, allows users to store data securely, back up their information, and record them. Different training resources can be virtualized and then transferred to educational clouds over the Internet. In this way, they provide teachers and students with an appropriate platform for information exchange. Scheduling computing tasks on VMs (Virtual Machines) based on predetermined purposes is a key consideration in cloud computing [3, 4]. The primary purpose of task scheduling is to reduce the time and energy required to complete tasks, enhance resource utilization, and balance workload. By shortening task completion time, large numbers of cloud users can then be served more quickly [5, 6]. This improves user experience, balances workloads, enables VMs to be fully utilized, and avoids resource idleness or decreased productivity that would otherwise occur [7–9]. But these two objectives are interrelated, for instance, the tasks can be scheduled based on resources with strong computing power for reducing task completion time, which in turn leads to the issue of lack of load balancing. Therefore, when designing and optimizing the algorithm, it is essential to maintain a balance between tasks completion times and load balancing. In computing, scheduling is to map multiple tasks to proper processors so that it could optimize one or more objectives at an acceptable time [10]. The problem is getting difficult when a large number of tasks assigned to a system at the same time that makes the problem of scheduling a complex problem to finding an optimal answer. The task scheduling problem could be modeled as a Traveling Salesman Problem (TSP) [11] or Vehicle Routing Problem (VRP) [12]. While both TSP and VRP are NP-hard problems, based on the proves given in [13], task scheduling is also a NP-hard problem. Recently, meta-heuristic methods [14–17], have highly been utilized to solve such optimization problems at an acceptable time. Most of these algorithms have been created by inspiration from physical processes or the animals' behavior and usually act collectively. The Gravitational Search Algorithm (GSA) is a new meta-heuristic algorithm that has been created based on the law of gravitation that could solve TSP type problems efficiently. This method, utilizes swarm intelligence to find patterns. In the problem space, the system space is determined, which includes multidimensional coordinates. During the creation of the system, governing rules are determined. In this paper, the problem of task scheduling in cloud computing is solved using the proposed GSA-based hybrid algorithm. Previous methods have some deficiencies in searching the problem space. Hence, a better algorithm in terms of discovery was needed to address this problem. Combining the Genetic Algorithm (GA and GSA algorithms can create a strong algorithm both for discovery and extraction, as GA is a global search algorithm and GSA is a Local Search Algorithm (LSA). This paper is organized as follows. A brief review of the related works followed can be found in Sect. 2. A detailed explanation of the research question appears in Sect. 3. Sect 4 presents a comprehensive description of the proposed GSAGA for solving the cloud computing task scheduling problem. Results and conclusions are presented in Sects. 5 and 6, respectively.

2 Related works

The purpose of this section is to present accomplished research in scheduling optimization. Tumbling task completion period and maximization of resource utilization in a cloud are some of the most superior issues in task scheduling [18]. To decrease the execution time of task scheduling a method based on Probabilistic Linguistic Term Set (PLTS) has been proposed in [19] by Zhao et al. [19]. Later, [20], presented an improved GA for task scheduling which groups VM and allocates the tasks to these VMs in a way that the task completion time is reduced and resource utilization is enhanced. Modified Breadth First Search (MBFS) was proposed in [21] to decrease the average execution and waiting times. Reducing the number of servers is also part of optimizing task scheduling. A task scheduling algorithm based on Vacation Queueing Models (VQMs) was presented by Cheng et al. [22] to reduce PM energy in clouds. According to Duan et al. [23], power consumption can be reduced by turning off idle PMs by using the Energy Aware Task Scheduling (EATS) process. A novel energy-aware task scheduling algorithm (EATS) was presented in Ismail et al. paper [24] on cloud computing. It was found that CPU power consumed the highest percentage of PM energy. Approximately 68 and 54% of a computer's total power is used by startup and shutdown procedures, respectively. Scheduling for server power on/off cycles should be avoided for energy-aware schedules. Network bandwidth availability has also been examined in that research. Lin et al. [25] developed a bandwidth-aware task scheduling (BATS) algorithm that shows how to allocate tasks for virtual machines as optimally as possible based on a nonlinear programming model. It reduces execution time significantly in cloud environments with limited bandwidth. In order to schedule tasks, Rasaque et al. [26] employed a workflow scheduling algorithm that took network bandwidth into account. With this method, resource consumption and output can both be significantly reduced. In addition to load balancing and QoS surveys, there has been research on load sharing. A task scheduling algorithm called MQoS-GAAC was developed by Dai et al. [26] based on Ant Colony Optimization (ACO) and Genetic Algorithms (GA). Performance of this algorithm is good because it balances resources and maintains QoS. Dynamic voltage and frequency scaling (DVFS) has been used in many studies to reduce energy consumption. [28] Tang et al. proposed an algorithm to schedule workflow tasks with DVFS-enabled energy efficiency to reduce energy consumption in PM processes by shutting down idle PMs (low voltage and frequency). Zhang et al. [29], estimated the Cloud Freq without knowing the constraint, which permitted the algorithm to improve the balance between schedule length and energy savings. This study employed DVFS technology to save energy by reducing CPU frequency, which in turn increased energy consumption, and at the same time increase the runtime of the cloud platform. In this regard, DVFC technology does not affect energy consumption in a significant way. As it was mentioned earlier, the use of meta-heuristic methods for solving complex optimization problems has recently been remarkable [29] since comparing to conventional methods with (Exponential-time complexity), these methods can obtain relatively optimal responses in polynomial times (polynomial-time difficulty/complexity). Indeed, meta-heuristic methods and various versions of them have been utilized for solving scheduling problems in various fields such as cloud

computing. According to the report in [29], the GA [30] and swarm intelligence algorithms like ACO [31] and Particle Swarm Optimization (PSO) [32] can be mentioned among the employed meta-heuristic methods in cloud task scheduling. These optimization algorithms have been adopted from biological population transformations and can solve complicated general optimization problems via cooperation and competition among the individuals [32]. In fact, the use of meta-heuristic algorithms in cloud computation is increasing and various meta-heuristic methods have been proposed such as scheduling algorithms based on GA, PSO, and ACO Aziza et al. [33] propose a GA system that shares time and space to reduce scheduling times and processing costs. Using the ACO algorithm, Li et al. proposed a Task algorithm for cloud computing tasks [34]. They developed a new PSO algorithm that balances the energy consumption of data-intensive services with the quality of overall service. A combination of these methods has been used in some surveys to resolve the task scheduling problem. The PSO-ACO method was created to be more efficient than unit algorithms for scheduling tasks by Chen et al. [35]. By applying the general searching capabilities of GA to ACO, Liu et al. [36] could initialize pheromones in the ACO style. A hyper-heuristic algorithm was created by combining GA, ACO, and PSO in a unified framework in edict to reduce the scheduling period. In this study, GA and the force of gravity are tested in a multi-purpose model that improves the efficiency of computing systems by using these methods. Table 1 summarizes a comparison of different task scheduling approaches.

2.1 The gravitational search algorithm (GSA)

The Gravitational Search Algorithm (GSA) has been inspired by the law of gravitation and by means of Newton's law of gravitation [37]. In this algorithm, the search agents are a group of masses. According to Newton's law of gravitation, each object exerts force on other objects and attracts them toward itself. Whatever these objects are bigger and closer, the effect of this force will be higher. Thus, each object recognizes the place and amount of mass of other objects using the force of gravity and this force can be used as a tool for information exchange based on the target function, the algorithm identifies the mass location in the problem space, as well as how many masses exist in the problem space. The GSA is used for solving optimization problems. In this algorithm, the search space is defined as a set of m particles. During the search, potential solutions are taken into consideration based on particle positions within search space. Equation (1) gives the particle's position x_i:

$$x_i = \left(x_i^I, ..., x_i^d, ...x_i^D\right) \tag{1}$$

Within this system of time, particles are exerting forces on one another in the direction of the dimension. Equation (2) computes the gravitational force based on *G(t), (t),* and *, in which G(t) is the gravitational constant in time. There are two masses associated with particles *i* and *j*, respectively. We have used Euclidean distance to find the reserve between particles (Eq. 3).

Table 1 Comparison of different task scheduling approaches

Authors	Year	Algorithm	Goal	Method / solution	Disadvantages
Zhao et al. [14]	2018	Prediction-based and locality-aware task Scheduling (Plts)	Prediction-based and locality-aware task scheduling for parallelizing video transcoding	It combines the benefits of two traditional heuristic scheduling algorithms, max–min and min–min, to make load balancing in cluster	Should further explore how to improve the prediction accuracy
Li et al. [15]	2016	Improved genetic algorithm	Task scheduling in mobile cloud computing	A task-virtual machine (VM) assignment strategy and improve genetic algorithm (GA) is presented	Stuck in the local optimal
Yadav et al. [16]	2014	MBFS algorithm	Task scheduler in cloud computing	Proposed an integrated task scheduling algorithm that takes into account the issues such as VM management and Datacenter management	Not having the characteristics such as memory and bandwidth be taken into consideration to prioritize the VMs
Duan et al. [18]	2017	Improved ant colony algorithm	Task scheduling in heterogeneous cloud computing systems	New scheduling approach named PreAntPolicy that consists of a prediction model based on fractal mathematics and a scheduler on the basis of an improved ant colony algorithm	Stuck in the local optimal
Dai et al. [20]	2015	Hybrid genetic algorithm and ant colony algorithm	Task scheduling in cloud computing	A novel task scheduling algorithm MQoS-GAAC with multi-QoS constraints considering the time-consuming, expenditure, security and reliability in the scheduling process	High execution time of the algorithm

Table 1 (continued)

Authors	Year	Algorithm	Goal	Method / solution	Disadvantages
Aziza and Krichen [27]	2017	Genetic algorithm	Task scheduling in cloud computing	The main role of model is to estimate the time needed to run a set of tasks in cloud and in turn reduces the processing cost	Stuck in the local optimal
Chen and long [29]	2019	Hybrid particle swarm algorithm and ant colony algorithm	Task scheduling in cloud computing	The proposed algorithm is capable of keeping particles in the fitness level at a certain concentration and guarantee the diversity of population	High execution time of the algorithm
Pirozmand et al. [10]	2021	Multi-objective hybrid genetic algorithm	Task scheduling in cloud computing	A two-step hybrid method for scheduling tasks aware of energy and makespan The first step involves prioritizing tasks, and the second step consists of assigning tasks to the processor	High execution time of the algorithm because of using a proposed meta-heuristic algorithm in multiple steps

$$F_{ij}^{d}(t) = \frac{G(t) \times M_i(t) \times M_j(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t) \right) \tag{2}$$

$$R_{ij}(t) = X_i(t), X_j(t)_2 \tag{3}$$

Based on Eq. 4, all the forces acting in the same direction on a particle are equal to or equal to the forces acting on it from all the other particles (in the direction of time). These are the quantities that make up this equation: radius r_j and force f_{ij}.

$$F_i^d(t) = \sum_{j=1, j \neq i}^{m} r_j f_{ij}^d(t) \tag{4}$$

Particles must accelerate in the same direction as they travel, and if a force is applied in that direction, the acceleration will be proportional to its mass. Equation (5) can be used to determine the particle's acceleration in time.

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{5}$$

A particle's velocity is determined by summarizing its present velocity and its acceleration, based on Eq. (6). Assuming the particle's current position and velocity are added, Eq. (7) computes the particle's new location.

$$V_i^d(t+1) = r_i \times V_i^d(t) + a_i^d(t) \tag{6}$$

$$X_i^d(t+1) = r_j \times X_i^d(t) + V_i^d(t+1) \tag{7}$$

The random numbers ri and rj are generated according to a uniform dissemination in the interlude *(0,1)* and are used to maintain the randomness of the search. For the adjustment of the gravitational constant, Eq. (8) is used. It declines exponentially when the gravitational constant is increased [27, 28].

$$G(t) = \beta^{-a\frac{1}{T}} \tag{8}$$

α is a positive constant, and T is the quantity of algorithm repetitions. In other words, T is the life of the system.

3 Problem statement

In cloud computing, task scheduling policies directly impact resource usage efficiency. On the basis of the independence of each task, we can summarize task scheduling in a cloud environment as follows: [38]

(1) Detailed information about input tasks and available resources (virtual machines) is necessary for mapping tasks and resources.
(2) Upon receiving the assigned requirements (i.e., optimization objectives), the task scheduler will generate an optimized task execution plan.
(3) Cloud computing ultimately delivers the most efficient plan for execution.

The main challenge in task is to optimally allocate some tasks to a given number of virtual processing machines. Therefore, research work has put the spotlight on task scheduling optimization of VMs [39]. Here, the following model is considered to demonstrate the detailed optimization process of the scheduler. There are N_t independent tasks defined by a length set, $T = \{T_1, T_2 \ldots, T_{N_t}\}$ is the length of the *nth* task in Million Instructions *(MI)*, where *Ti* is the length of the *nth* task in Million Instructions *(MI)*. Furthermore, the execution rates of *Nv* number of *VMs* are defined as $V = \{V_1, V_2 \ldots V_{Nv}\}$ where *Vj* represents the *j*th *VM* execution rate in Million Instructions Per Second (MIPS). Using $Nt > Nv$, a matrix *A* can be used to determine which task should be assigned to which *VM*, as follows.

$$A_l = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1Nv} \\ a_{21} & a_{212} & \cdots & a_{2Nv} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N_t 1} & a_{N_t 2} & \cdots & a_{N_t Nv} \end{bmatrix}, \text{ where } \quad a_{\text{ij}} = \begin{cases} 1 \text{ if } T_i \text{ is allocated to } V_j \\ 0 \; f \; T_i \text{ is allocated to } V_j \end{cases}$$

We consider CPU computing power and memory size of every resource node. In addition to these two attributes, resource bandwidth is employed to abstract the general recourse that a node can provide. It is defined as a function of the other two attributes. That is to say, the bandwidth is defined by CPU power and memory size [38]. While C_n, M_n, and L_n are used to model the cloud computing system, C_t, M_t, and L_t can be used to model each computing task. Table 2 indicates notations used to facilitate traceability during the research process. A schedule is determined by the optimization objectives and the feasibility constraints. The

Table 2 Notations used in task scheduling model

Notation	Definition
N_t	Tasks assigned at arrival
Nv	The number of VMs
a_{ij}	Allocation matrix to demonstrate assignment of ith task to the jth VMs
T	Set of arrival tasks
C_n (C_t)	Processing capability vector for VMs (tasks)
M_n (M_t)	Load capability vector for VMs (tasks)
L_n (L_t)	Resource bandwidth vector for VMs (tasks)
Wi	Weight of each cost function
P	Price unit: through a single simulation with low percent capacity mean load, a static and per-unit price of computation is provided

purpose of an optimization problem that has several objective functions is to find a trade-off solution that represents the best compromise between all of those functions [40, 41]. According to our constraints on CPU, memory, and resource bandwidth, we define three objective functions, namely the time cost function f_1, the load cost function f_2, and the price cost function f_3 represented by (9), (10), and (11), where $C_{\mathrm{t,i}}$ revenue the C_t significance of the ith task and $C_{\mathrm{n,j}}$ is the C_n importance of jth VM. Similar demonstration is also considered for the signs M and L. In this work, we choose the price cost to simplify the resource cost. Therefore, we use a price unit P

$$f_1 = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} \frac{\mathrm{C_{t,i}}}{\mathrm{C_{n,j}}} \tag{9}$$

$$f_2 = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} \frac{\mathrm{M_{t,i}}}{\mathrm{M_{n,j}}} \tag{10}$$

$$f_3 = \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} \frac{\mathrm{C_{t,i}}}{\mathrm{C_{n,j}}} \times \frac{\mathrm{L_{t,i}}}{\mathrm{L_{n,j}}} \times P \tag{11}$$

The reason why we added such a factor $\frac{M_{\mathrm{t,i}}}{M_{\mathrm{n,j}}}$ for each task while calculating the total cost in f_3 can be attributed to one feature of a computing system according to which the cost depends not first on the task implementation time but also on the relation of resource usage at individually time point. We have a Multi-Objective Optimization (MOO) problem here. These functions have different and often completely at odds goals, so we want to minimalize the values of these functions. For a task to be processed more quickly, a powerful processor would be required while this would also increase the cost. The min–max normalization method is used to solve such MOO problems. Due to the different symbols in the problem, A, B, and C, this prevents the search path for an optimal solution from skewing. Three normalized objective functions represented as F_1, F_2, and F_3 are displayed below.

$$F_1 = \frac{1}{\mathrm{N_t}} \sum_{i=1}^{\mathrm{N_t}} \sum_{j=1}^{\mathrm{Nv}} a_{ij} \frac{\mathrm{C_{t,i}} \,/\, \mathrm{C_{n,j}}}{\max_{\forall i,j} \left\{ \mathrm{C_{t,i}} \,/\, \mathrm{C_{n,j}} \right\}} \tag{12}$$

$$F_2 = \frac{1}{\mathrm{N_t}} \sum_{i=1}^{\mathrm{N_t}} \sum_{j=1}^{\mathrm{Nv}} a_{ij} \frac{\mathrm{M_{t,i}} \,/ \mathrm{M_{n,j}}}{\max_{\forall i,j} \left\{ \mathrm{M_{t,i}} \,/ \mathrm{M_{n,j}} \right\}} \tag{13}$$

$$F_3 = \frac{1}{\mathrm{N_t}} \sum_{i=1}^{\mathrm{N_t}} \sum_{j=1}^{\mathrm{Nv}} a_{ij} \frac{(\mathrm{P\ C_{t,i}\ L_{t,i}}) / (\mathrm{C_{n,j}\ L_{t,i}})}{\max_{\forall i,j} \left\{ (\mathrm{P\ C_{t,i}\ L_{t,i}}) / (\mathrm{C_{n,j}\ L_{t,i}}) \right\}} \tag{14}$$

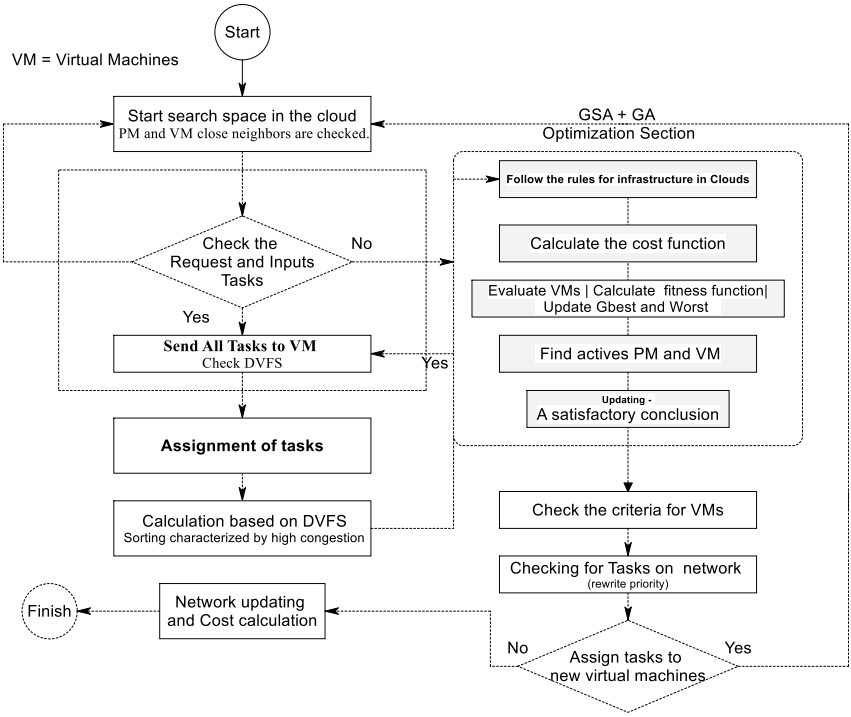

Fig. 1 The GSAGA method is illustrated

Some weight values are employed to meet desperate requirements of different cloud computing systems, and make our target function adjustable. Following is an illustration of the ultimate objective function for optimization:

$$F_{\text{opt}} = \min\left\{w_1F_1 + w_2F_2 + w_3F_3\right\} \tag{15}$$

We aim to minimize the value of ($w_1\ F_1 + w_2\ F_2 + w_3\ F_3$) under an optimization algorithm. As a result, w_i will be affected by the size of W (weight of cost function), so the algorithm will prioritize reducing F_{i} to a low value. In this paper, for the simplifying the implementation process, we simply consider $w_1 = w_2 = w_3 = 1/3$ for a general case so that our optimizations on the TSP in a clouds system will be as follows:

$$F_{\text{opt}} = \min\left\{\frac{1}{3}\left(F_1 + F_2 + F_3\right)\right\} \tag{16}$$

4 The proposed algorithm (GSAGA)

The stages of the GSAGA are as below (Fig. 1).

(1) The user sends their request to the manager.
(2) The manager sends the requests to resources.
(3) Given the GSA and the GA fitness function, some operations are performed on the resources.
(4) Using the GA fitness function, some of the most optimal resources are selected.
(5) The optimal resources have been selected.
(6) The cost is improved and the energy is reduced for the selected optimal resources.

There are a large number of defined probable solutions for NP particles in a d-dimensional search space. The dimensions have been determined by using virtual machines. Using these probabilities, we use the worst and best cases. The fitness value of each particle has been calculated based on the performance of the fitness function. Situation of dimension d with the mass i has been shown with np_i^d.

$$NP_i = \left(np_i^i, np_i^2, \ldots\ldots\ldots, np_i^n, \ldots\ldots\ldots, np_i^d\right) \tag{17}$$

Various solutions are assumed to be possible within a multidimensional search space. The solution to the problem is found in each point in space, and each solution has a "mass" which determines the objective function. More objective value can be generated by producing better solutions, thereby increasing their mass. Search agents themselves are also particle collections. It is assumed that gravitational and motion laws govern the space we construct as a search space [21, 23]

4.1 Fitness function

This method is used for searching in the problem space that is carried out for finding the superior but not necessarily the optimal response. The GA can be introduced as a general search method that imitates the laws of biological evolution. A GA is an exploratory search that imitates natural selection. This search (that is sometimes recognized as a processing) is usually employed to create helpful solutions for optimization and searching of problems. In this method, each chromosome is considered as a VM. If $S = \left\{VM_1, VM_2, \ldots, VM_n\right\}$ is regarded as a set of chromosomes, the chromosome which has the greatest chance for selection can be selected using the ranking method. In the selection scheme based on the ranking method, chromosomes are stored for the first time given the fitness values of their organs in the population. The selection scheme of chromosomes is via Eq. (18)

$$p_i = \frac{R(i)}{\sum_{i=1}^{n} R(i)} i = 1, 2, \ldots, n \tag{18}$$

In this formula, p_i shows probability of choosing chromosome i, n shows the population size and is the rank of chromosome i. In this case, $R(n)$ offers the best chromosome and $R(1)$ is the worst chromosome.

4.2 Incurred cost

It is notable that the total cost of using the resources should be minimized. Assume that R_j indicates a unit price for resource j that is fixed and maximum cost is the biggest cost arising from a user on resource j. Hence, the cost to execute task i can be calculated by means of Eq. (19)

$$F_{\text{cost}}(I) = \frac{T(i,j) \times R_j}{\text{Max cost}^{1-a}} \tag{19}$$

In the above relation, maximum cost is the most expensive task. Therefore, the total cost of a solution (chromosome) that indicates the cost of one chromosome in the population can be illustrated via Eq. (20)

$$F_{\text{cost}}(B) = \sum F_{\text{cost}}(j) 1 \geq j \geq M \tag{20}$$

4.3 Energy savings

In this section, it is noteworthy that the energy model is one of the energy-saving methods at the system level; therefore, the below method is conducted to reduce energy at the system level. Dynamic voltage scale acts according to a simple principle that decreases the power supply voltage as well as the clock frequency to CPU to spend less energy. To do this, an energy model that is obtained from the energy consumption model in Complementary MOs under the model of dynamic power, power of the processor, and dynamism is utilized which is achieved using Eq. (21).

$$P_{\text{daynamic}} = \text{AC}_{\text{ef}} V^2 f \tag{21}$$

In the above relation, A is the number of switches for each clock cycle (time),C_{ef} shows the effective capacity of load, V shows the power supply voltage and f shows the operating frequency. Equation (21) indicates that the source voltage is an important and major criterion. Hence, its decreased value will be more effective on decreased value of energy consumption. Capacity is computed based on Eq. (22)

$$\text{Capacity} = \text{MIPS} * \text{PE} + \text{BW} \tag{22}$$

According to [42], the power–frequency relation at any level of efficiency can be estimated via a second-degree model, i.e. it can be written that:

$$P(f) = P_{\min} + \alpha \left(f - f_{\min}\right)^2 \tag{23}$$

In Equation, power *(P)* is in terms of *(W)*, frequency *(f)* is in terms of CHz and α is a coefficient in terms of W/(GHz)^2. It has been assumed in the model that the host CPUs in a data center follow such a nonlinear relation. CPUs are executed with a limited number of executive frequencies in the range $[f_{\min}, f_{\max}]$. This leads to a trade-off between efficiency and the costs related to power. In this model, all servers or hosts follow this relation despite the fact that they may provide the value of a different CPU

resource. Moreover, α will be in terms of W/(GHz)^2, because the highest level of consumption power is when the CPU is executed with the highest power. Considering this issue, α is calculated via Eq. (24)

$$\alpha = \frac{P_{\max} - P_{\min}}{\left[\left(f_{\max} - f_{\min}\right)\right]^2} \tag{24}$$

Moreover, the hosts or servers in the data center can be inactive if the load over the data center is low. For using the proposed model, there must be coordination among various VMs for the complete updating of all DVFSs [42]. The use of GSA + GA and allocation of the set of successive tasks to a physical machine can be effective on QoS increase.

4.4 Updating of masses on the basis of an evaluation function

The best situation of particles is returned to the tasks for load scheduling. Then, they are allocated to the related VMs in data centers based on the situations for execution. Afterward, total scheduling of the cloud GSA continues until all tasks are executed in the VMs in the cloud. Thus, minimum time and minimum cost of total calculations are obtained. Fitness represents the minimum value among particles, i.e., the fitness value. In the search space, the worst value represents the maximum fitness. New situations are regarded as the place of new masses in the search space. The weight of new masses is normalized via the below equations. Values of best and worst for particles are calculated as below to minimize total cost and mass *Mi(t)*.

$$m_{\mathrm{i}}(t) = \frac{\mathrm{fit_j(t)} - \mathrm{worst(t)}}{\mathrm{best\ (t)} - \mathrm{worst\ (t)}} \tag{25}$$

$$M_{\mathrm{i}}(t) = \frac{m_{\mathrm{i}}(t)}{\sum_{j=1}^{N} m_{\mathrm{j}}(t)} \tag{26}$$

4.5 Values for best and worst mass

In this section we calculate and update the strongest and weakest values for best and worst Mass. In the following Equations, $fit_j(t)$ demonstrates the value of goodness of mass i in time t and worst and best indicate competency of the worst and the best population factor across time. They are calculated using the following relations in minimization problems:

$$best(t) = \frac{min}{j \in (1, \ldots N)} fit_j(t) \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{27}$$

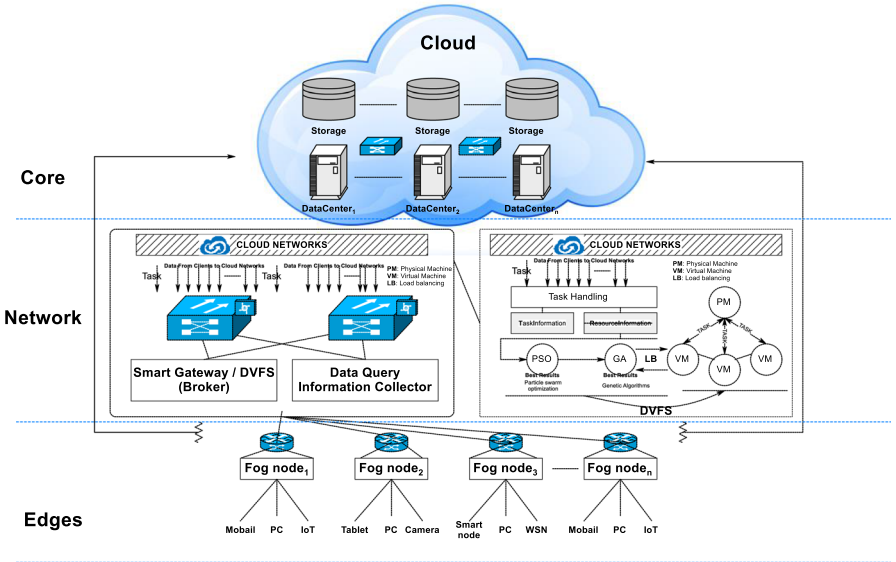

Fig. 2 Cloud structure and edge technology architecture

$$\text{worst}(t) = \frac{max}{j \in (1, \ldots N)} fit_j(t) \tag{28}$$

The GSAGA in this paper is inside the cloud environment. When the user sends a request to the manager, the manager receives the request and sends it to the individual VMs. Thus, this force is used as a tool for exchange of information. Newton's law of universal gravitation has a major role in finding the best optimal response among the objects (VMs). Here, the best and the most optimum VMs are selected using the GA fitness function. Then, the population recognizes some children as active children because their chromosomes have been optimized using the GSA. Repeatedly, chromosomes are chosen and the GSA is recalled for them. In this way, energy and cost are reduced for the obtained active VMs.

5 Simulation and experimental results

While designing a system for the cloud data center on a large scale, various experiences and tests must be executed on a real substructure. This allows for the evaluation of different algorithms. In the simulation process, various entities are created, and they all test under different tasks, exchanging messages or information if needed. In this section, we present detailed simulation results and comparisons with other algorithms (Fig. 2). To simulate the GSAGA in a cloud environment, we use CloudSim 3.3 and the NetBeans is chosen to employ in the Integrated Development Environment (IDE) of this model. The CloudSim simulator is an OpenSource

software developed using Java through which cloud computing environments can be modeled and the efficiency of applied services can be tested [42, 43]. Namely, the entities in which it is defined communicate with each other via sending the events. The simulation scenario of the proposed algorithm is shown in Fig. 3. Among Simjava's classes, the entity class operates on all the entities (GSA + GA, DVFS) that are to be developed. These are the sub-classes of this class. During simulation, each established entity in this software is executed as a self-regulating line. Each event class represents each exchanged message between the entities. An additional method is used for executing multiple entities at once to increase efficiency. It is possible to queue a set of entities and recall them in the order of execution time in CloudSim. We first explore SLA, Makespan, and Gain for various tasks in the Simulation Results Section. Cloud-based simulations have been used to perform Epigenomics work. The GSAGA has been compared with scheduling algorithms based on both CLS and SLA including Profit-MCT [43], SLA-MCT [43], the simulation approach based on the Whale Optimization Algorithm (WOA) for scheduling of tasks in cloud computing, and the improved WOA for cloud task scheduling (IWC) [38] algorithms. When the requested VMs of an application is determined on separate hosts and started to execute the program, the output or responses of programs in each VM is considered to offer a secure service in certain periods. The comparator receives the outputs of each VM as input and compares them pairwise. In fact, we have developed the fundamental of this emulator to model the GSA + GA as well as

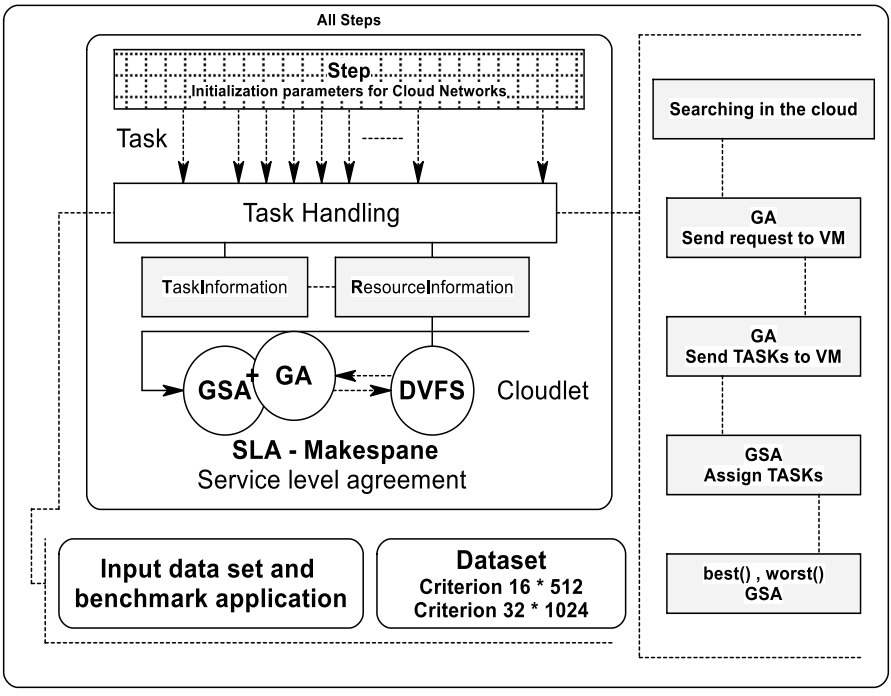

Fig. 3 Simulation scenario

the effect of DVFS tools on Clouds and calculation of the added overhead [43]. In the following Subsections, some cases such as the efficiency of the processor, workflow overhead, energy consumption of physical machines, and data locality will be explored for various tasks.

5.1 Datasets and benchmarks

In this subsection, the algorithms are presented with the input dataset. Cloudlets are a concept in Cloudsim that illustrates a workload (the workload can include input–output and processor operations), which can be used to establish algorithms. There are a few parameters that describe a Cloudlet, including its ID number, million instructions per second, number of processor cores, input and output file sizes, processing time, and models of memory efficiency, bandwidth efficiency, and processor efficiency. Three different sizes of datasets with twelve samples in every one of them were considered in the simulation procedure. These twelve samples demonstrate the time required to complete a task in a cloud depending on available uniform distributions in the dataset. Twelve samples of each dataset are separated based on the below factors: (1) Type of compatibility *(x)*, (2) Work heterogeneity *(yy)*, and (3) Cloud heterogeneity *(zz)*. The main structure of these samples is u_{xyyzz}. It is mandatory to determine the processing time and instructions per second randomly in the above-mentioned scenarios for the simulation of the cloud environment. Therefore, one sample of normal distribution function should represent the processing size of Cloudlet for GSA + GA, while the sample of Poisson distribution should represent the processing time. In the present study, the mean and variance of the normal distribution function are considered as 40,000 and 10,000, respectively ($\sigma = 10{,}000$, $\mu = 40{,}000$), for the estimation of million instructions of each Cloudlet. Table 3 shows the specifications of input tasks for simulation. A normal distribution is the probability density function of Eq. (29) and a Poisson distribution is the probability mass function of Eq. (30). The instruction rate is therefore related to this distribution function. The cumulative distribution function can be reversed to create pseudorandom numbers in this case. VM processing power and capacity are presented in Table 4 for a limited sample of VMs. In this study, we assume that 100 workloads enter the cloud on average in each time unit (λ-100). To prevent mathematical complexities in this simulation, the ColtPackage in Java has been used. Moreover, Table 5 displays an incomplete sample of machines and their processing power and capability that are for data sets 512 and 1024, and cases such as hypervisor bandwidth of the VM, the volume of the VM, and processing elements have been taken into account [41].

$$f\left(x;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\pi)/2\sigma^2)}, x \in \mathbb{R} \tag{29}$$

Table 3 Specification of input tasks for simulation

Arrival time	Dataset	MIPS
The random number of the inverse of Poisson cumulative distribution function with value = 100	The main structure of these examples is u_{xyyz} For 16 * 512 and 32 * 1024 modes	The random number of normal distribution function with a mean of 40,000 and standard deviation of 10,000

Table 4 Here are a few examples of virtual machines in the form of virtual machines with various processing power and power consumption

ID	Type	MIPS	RAM	Bandwidth	VMs	Virtual machine volume
1.1	A.A	300	512 MB	[100,1000] Mb/s	Xen	2200 MB
1.2	B.B	500	256 MB	[100,2000] Mb/s	Xen	2200 MB
1.3	C.C	300	512 MB	[100,3000] Mb/s	Xen	2200 MB
1.4	A.C	500	256 MB	[100,4000] Mb/s	Xen	2200 MB
1.5	A.B	300	512 MB	[100,5000] Mb/s	Xen	2200 MB
1.6	B.C	500	256 MB	[100,6000] Mb/s	Xen	2200 MB

Table 5 Here is an example of how powerful and fast a virtual machine

ID	Type	MIPS	RAM	Bandwidth	VMs	Virtual machine volume	Processing elements	Dataset
1	A	i * 5 mod 301	512 MB	100 Mb/s	Xen	2200 MB	GSA + GA	Criterion 16 * 512
2	B	i * 5 mod 501	256 MB	100 Mb/s	Xen	2200 MB	GSA + GA	Criterion 32 * 1024

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{30}$$

In the current study, there are 25 virtual machines and 50 virtual machines. Variables for virtual machines include the ID of the physical machine, the number of processing elements (the number of processor cores), how much RAM is allocated, how much bandwidth each virtual machine has, and how they are scheduled. A VM is hosted on a physical machine, so these are considered to be the center of data because of their high processing power and capacity.

5.2 Experimental analysis of exploitation of processors

In Fig. 4, the amounts of CPU processing are similar as the number of tasks increases. In the GSAGA, the efficiency of the processor performed better compared with the CLS-based methods because of finding the best state by GA. All data centers are involved in processing since all data centers use the processor since the GSAGA uses GSA and updates the evaluation function. As a result of using only one data center in the Profit-MCT method, the processor's efficiency is comparable to the GSAGA. When the scenario was performed in a group of centers, the GSAGA improved to 4.12% in exchange for the mean of tasks.

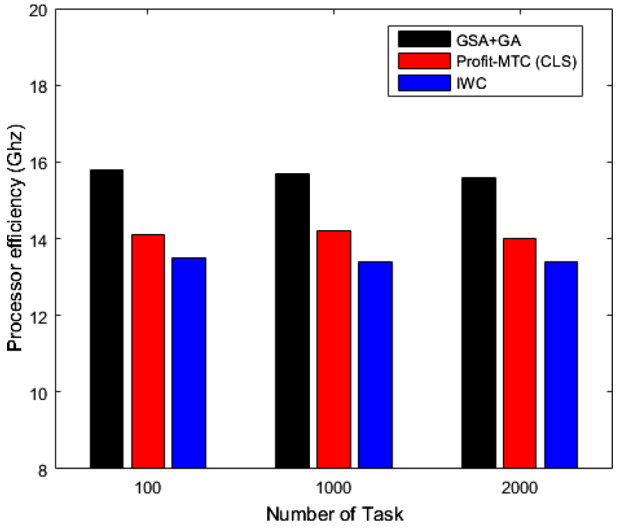

Fig. 4 Comparison of CPU efficiency against the number of tasks in Epigenomics mode

Compared with the IWC algorithm, the efficiency of the processors in scheduled tasks has been reduced by the increased number of tasks. This is due to the spatial complexity of finding a suitable path for a group of whales. As the GSAGA has a low cost, the efficiency of tasks is better compared with other methods because of the lower cost of the proposed method. In the GSAGA, updating of the strongest and the weakest mass, i.e., best and worst indicates the competency level of the best and the worst population factor in time and this is effective on the efficiency of processors in various tasks. Newton's law of universal gravitation has a major role in finding the best optimal state among VMs; hence, the manager has received the request for the best optimal state and has sent it to the machines that are used as a tool for information exchange.

5.3 Experients on benchmarks 512 and 1024

In this section, the GSAGA has been compared with Profit-MCT, SLA-MCT, and IWC Algorithms using the dataset 512×16 of benchmark [3]. Via the use of benchmark for 512×16, changes of makespan for various tasks have been investigated through benchmark (512×16) [44]. It is observed in Fig. 5, the GSAGA and other methods are subjected to some changes by the increased number of tasks in various states. Since various states for the GSAGA are better optimized than other methods, therefore, the GSAGA is more optimal. Changes in makespan in different tests are due to the use of DVFS and updating of the network based on the changes in voltage. In some increasing states, load balancing changes are also observed. Like other methods under comparison, the GSAGA has been subjected to changes in different states and has improved to 7% in comparison with Profit-MCT, 11% in comparison with SLA-MCT, and 5.3%

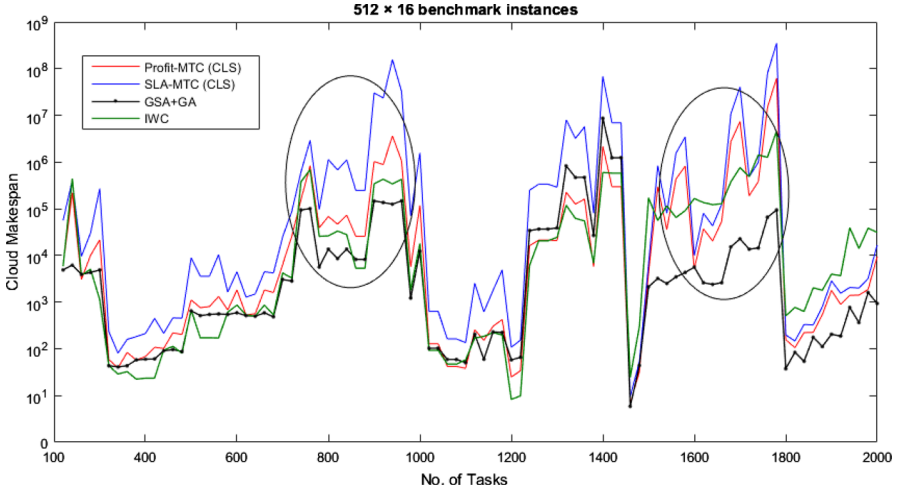

Fig. 5 Makespan comparison against task changes in the 512×16 benchmark

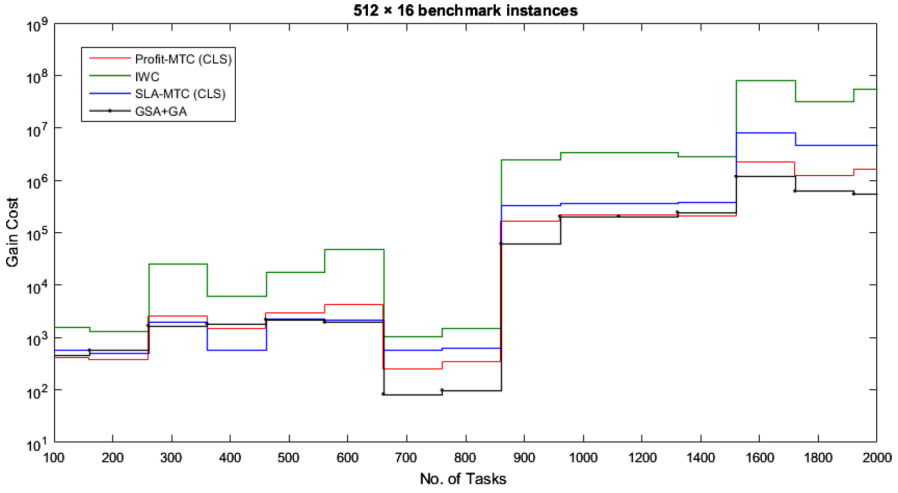

Fig. 6 Comparison of Gain Cost against task changes in the 512×16 benchmark

in comparison with IWC. In Fig. 6, changes in Gain Cost have been explored for various tasks and the cost of Gain Cost has been compared for various tasks in benchmark 512×16 [44]. As it is observed, the GSAGA has a lower cost than other methods. In SLA-MCT and IWC methods, changes in cost are similar to the performance of the GSAGA and it is better optimized (equal to 11.2%) than CLS in case of heavy workload (in case of load unbalance) and 6.17% in case of load balancing (load balancing using DVFS). In this case, the cost is increasing for all methods by increasing the number of tasks, and the GSAGA has less delay

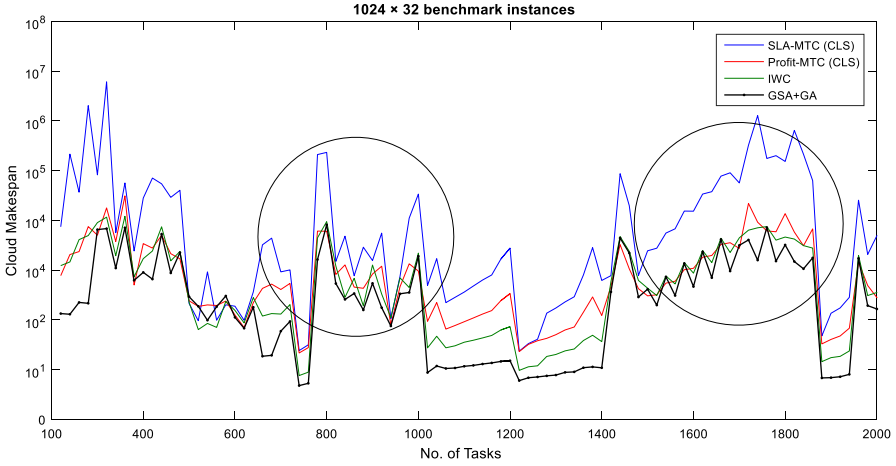

Fig. 7 Comparison of Makespan against changes in tasks in the 1024 × 32 benchmark

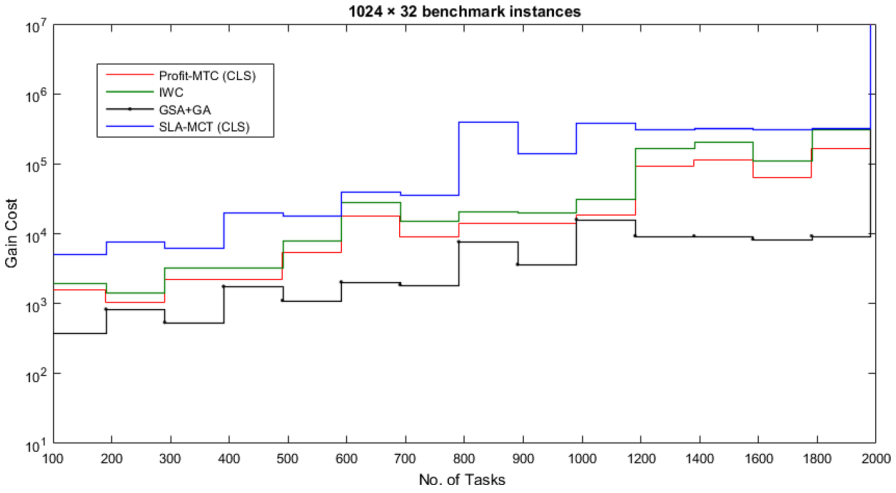

Fig. 8 Comparison of Gain Cost against task changes in the 1024 × 32 benchmark

than the other two methods. When the number of tasks is between 900 and 2000, the proposed GSAGA has better conditions than other methods. In the GSAGA, the execution of tasks is easier due to the best situation of VMs and suitable load balancing on physical machines. Then, the tasks are allocated to the related VMs to execute the instructions in data centers. Finally, total scheduling of GSA + GA continues until all tasks are executed on the VMs in the cloud to obtain minimum time and minimum cost of total calculations. Here, we compared the proposed GSAGA with Profit-MCT, SLA-MCT, and IWC methods using benchmark 1024 × 32 [44]. As it is observed in Fig. 7, changes in makespan for various tasks

have been explored in benchmark 1024. Because of the use of GA and GSA in a distributed form in the GSAGA, it has a lower makespan than other methods. It is better optimized than Profit-MCT (that is equal to 6.7%), SLA-MCT (that is equal to 9.3%), and IWC (that is equal to 4.2%). Changes in load and updating a network from the DVFS aspect (active mode and standby) are observable in tasks' slots. Changes in Gain for various tasks are explored in Fig. 8. The GSAGA has a lower cost than other methods. In SLA-MCT and IWC methods, changes in cost are similar to the performance of the GSAGA, and the degree of superiority of the GSAGA over IWC is equal to 8.23 where there is a heavy workload and 5.44 in load balancing (load balancing using DVFS). As it is observed, the cost for all methods is increased by increasing the number of tasks and the GSAGA of GSA + GA has less delay than IWC and CLS-based methods. The best situation of virtual machines with DVFS has been considered for active mode in the network and then, GSA + GA is effective on reduced complexity and cost. To explore cost, tasks are allocated to VMs to be executed in data centers. Finally, total scheduling of GSA + GA continues until all tasks are executed on virtual machines in a cloud to obtain minimum time and minimum cost of total calculations.

5.4 Data locality and workflow overhead experiments

Figures 9 and 10 shows experimental results on data locality for various tasks. Data locality means the transfer of calculations instead of data transfer for saving bandwidth. Data locality minimizes network traffic and enhances the total

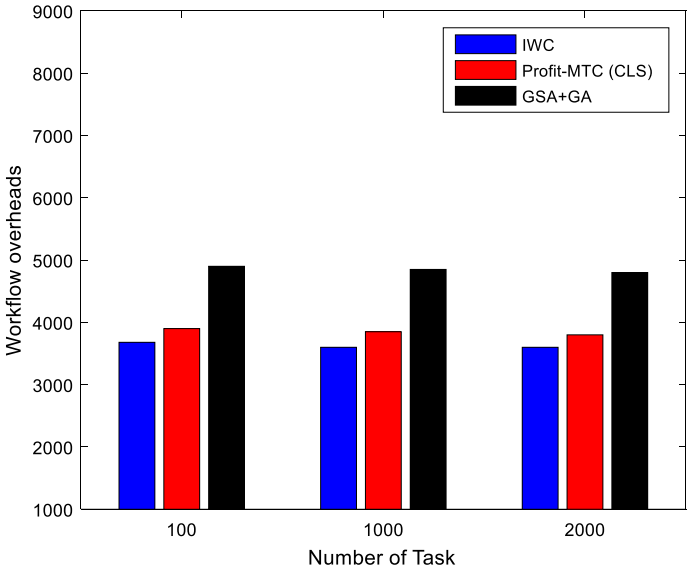

Fig. 9 A comparison of workflow overhead and epigenomics tasks

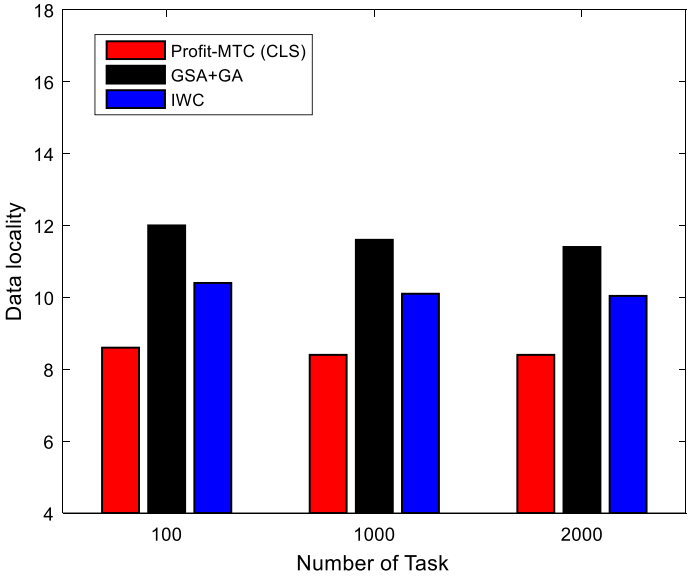

Fig. 10 In Epigenomics mode, compare the number of tasks with the locality of the data

performance of the system. Considering the use of clustering and Epigenomics flow in the GSAGA, it is observed that data locality has been optimal in HR state with Epigenomics flow, and processing operations have been performed successfully. In the GSAGA, considerable changes have not occurred by the increased number of tasks. In algorithms Profit-MCT and IWC which have complexity, involvement of the processor has been increased. IWC method has been optimal where no benchmark is used. A comparison is made between the GSAGA and the Profit-MCT and IWC methods concerning workflow overhead. In Epigenomics flow, we use two clustering factors: non-clustering and clustering. With the use of the database, the overheads exerted by the GSAGA and Profit-MCT method are optimal, with changes in overhead tasks leading to optimization. On the other hand, the IWC algorithm has more overhead and consumes more time.

5.5 Experimental analysis of energy consumption

In this section, the energy consumption of physical machines is explored. Figure 11 shows that the GSAGA has reduced energy consumption when compared to Profit-MCT and IWC methods, and has performed more optimally with Epigenomics flow. Since incremental tasks result in more complexity in output analysis, the values of tasks for the processing time have been presented incrementally. Our approach is to treat the number of virtual machines and workloads as integral coefficients. The energy consumption of physical machines has been measured in all data centers through a variety of tasks. Hence, energy consumption operations are performed in all data centers. The SLA-MCT and

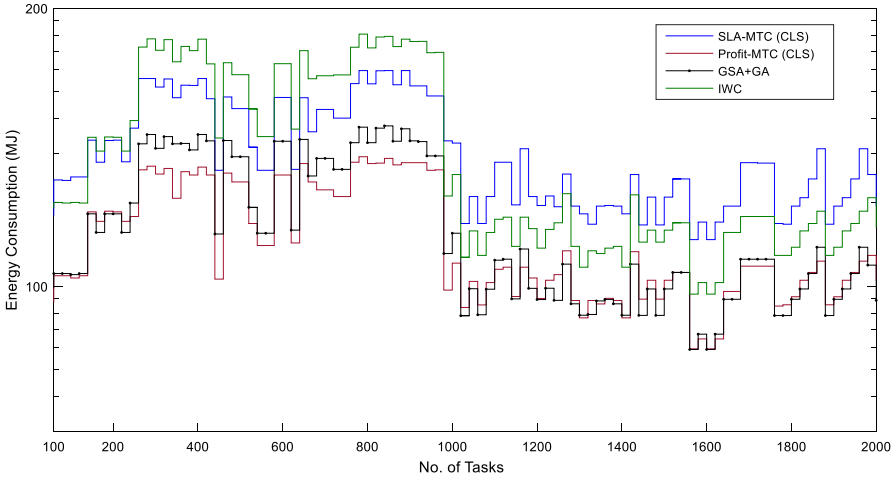

Fig. 11 Comparison of energy consumption per task for physical machines

Profit-MCT algorithms involve more processors and more time, respectively. The GSAGA is an optimal energy-consumption approach using DVFS in this workflow. Each algorithm has been similar to DVFS. Up to 2000 tasks, the GSAGA method is more reliable than Profit-MCT when the number of accomplished tasks is increased. When the number of VMs is low in relation to the load, GSAGA will perform better than IWC. Virtual machines exert a relatively low load in cloud environments and data centers offering cloud services. For this reason, the GSAGA will act better in the cloud environment. Like other methods under comparison, the GSAGA has been subjected to changes in different states and has improved to 7% in comparison with Profit-MCT, 14.3% in comparison with SLA-MCT, and 7.43% in comparison with IWC.

6 Conclusion

In this paper, we study the scheduling problem between tasks and resources. Scheduling is one of the most important aspects of improving cloud-based service productivity. Researchers have found that cloud computing has scheduling problems because many users have access to the cloud service provider. As a result, scheduling is an important component of cloud computing systems. There are several types of task scheduling algorithms that prioritize service quality and performance. A hybrid collective intelligence algorithm called GSAGA is presented in this paper. The GSAGA aims to analyze the VM analysis system, evaluate the run time, and calculate energy consumption, gain cost, and makespan. Various examples of virtual machines and tasks have been used to evaluate the GSAGA. The simulation results show that the GSAGA has appropriate and acceptable performance compared to the compared algorithms. Further, we suggest that for future work the other authors can use driven container technology (microservices

or serverless computing) for cloud applications and also SDN based network for improving the QoS [45].

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Panda SK, Jana PK (2015) Efficient task scheduling algorithms for heterogeneous multi-cloud environment. J Supercomput 71(4):1505–1533
2. Xiong N, Vasilakos AV, Wu J, Yang YR, Rindos A, Zhou Y, Song W-Z, Pan Y (2012) A self-tuning failure detection scheme for cloud computing service. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium: 2012: IEEE; pp 668–679
3. Javadpour A, Wang G (2022) cTMvSDN: Improving resource management using combination of Markov-process and TDMA in software-defined networking. J Supercomput 78:3477–3499. https://doi.org/10.1007/s11227-021-03871-9
4. Zhang F, Cao J, Li K, Khan SU, Hwang K (2014) Multi-objective scheduling of many tasks in cloud platforms. Futur Gener Comput Syst 37:309–320
5. Javadpour A, Wang G, Rezaei S (2020) Resource Management in a Peer to Peer Cloud Network for IoT. Wireless Pers Commun 115:2471–2488. https://doi.org/10.1007/s11277-020-07691-7
6. Yin Y, Chen L, Xu Y, Wan J, Zhang H, Mai Z (2020) QoS prediction for service recommendation with deep feature learning in edge computing environment. Mob netw and appl 25(2):391–401
7. Mirmohseni SM, Tang C, Javadpour A (2020) Using markov learning utilization model for resource allocation in cloud of thing network. Wireless Pers Commun 115:653–677. https://doi.org/10.1007/s11277-020-07591-w
8. Gao H, Huang W, Yang X, Duan Y, Yin Y (2018) Toward service selection for workflow reconfiguration: an interface-based computing solution. Futur Gener Comput Syst 87:298–311
9. Yin Y, Xu Y, Xu W, Gao M, Yu L, Pei Y (2017) Collaborative service selection via ensemble learning in mixed mobile network environments. Entropy 19(7):358
10. Pirozmand P, Hosseinabadi AAR, Farrokhzad M, Sadeghilalimi M, Mirkamali S, Slowik A (2021) Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. Neural Comput Appl 33(19):1–14
11. Rostami AS, Mohanna F, Keshavarz H, Hosseinabadi A (2015) Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. Appl Math Inform Sci 9(2):1–11
12. Hosseinabadi AAR, Vahidi J, Balas VE, Mirkamali SS (2018) OVRP_GELS: solving open vehicle routing problem using the gravitational emulation local search algorithm. Neural Comput Appl 29(10):955–968
13. Pinedo ML: Scheduling, vol. 29: Springer, 2012
14. Mirmohseni SM, Javadpour A, Tang C (2021) LBPSGORA: create load balancing with particle swarm genetic optimization algorithm to improve resource allocation and energy consumption in clouds networks. Math Problem Eng 29(10):955–968
15. Pirozmand P, Sadeghilalimi M, Hosseinabadi AAR, Sadeghilalimi F, Mirkamali S, Slowik A (2021) A feature selection approach for spam detection in social networks using gravitational force-based heuristic algorithm. J Ambient Intell and Hum Comput. https://doi.org/10.1007/s12652-021-03385-5
16. Peng Z, Rastgari M, Navaei YD, Daraei R, Oskouei R J, Pirozmand P, Mirkamali SS (2021) TCD-ABCF: A trust-based community detection using artificial bee colony by feature fusion. Math Probl Eng 2021:1–19. https://doi.org/10.1155/2021/6675759
17. Peng Z, Jabloo MS, Navaei YD, Hosseini M, Oskouei RJ, Pirozmand P, Mirkamali, (2021) An improved energy-aware routing protocol using multiobjective particular swarm optimization algorithm. Wireless Commun Mob Comput. https://doi.org/10.1155/2021/6675759

18. Zhao H, Qi G, Wang Q, Wang J, Yang P, Qiao L (2019) Energy-efficient task scheduling for heterogeneous cloud computing systems. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City, IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEE, pp 952–959
19. Zhao H, Zheng Q, Zhang W, Wang J (2016) Prediction-based and locality-aware task scheduling for parallelizing video transcoding over heterogeneous mapreduce cluster. IEEE Trans Circuits Syst Video Technol 28(4):1009–1020
20. Li J, Li X, Zhang R (2016) Energy-and-time-saving task scheduling based on improved genetic algorithm in mobile cloud computing. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing Springer, pp 418–428
21. Yadav R, Kushwaha V (2014) An energy preserving and fault tolerant task scheduler in cloud computing. In: 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014), IEEE, pp 1–5
22. Cheng C, Li J, Wang Y (2015) An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing. Tsinghua Sci Technol 20(1):28–39
23. Duan H, Chen C, Min G, Wu Y (2017) Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. Futur Gener Comput Syst 74:142–150
24. Ismail L, Fardoun A (2016) Eats: Energy-aware tasks scheduling in cloud computing systems. Procedia Comput Sci 83:870–877
25. Lin W, Liang C, Wang JZ, Buyya R (2014) Bandwidth-aware divisible task scheduling for cloud computing. Softw Pract Exp 44(2):163–174
26. Shankar Eappen T, Abttan RA, Hassan F, Venugopal K (2018) List of contents. Inter J Eng Technol 7(4):124
27. Dai Y, Lou Y, Lu X (2015) A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing. In: 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IEEE, pp 428–431
28. Tang Z, Qi L, Cheng Z, Li K, Khan SU, Li K (2016) An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. Journal of Grid Computing 14(1):55–74
29. Zhang Y, Wang Y, Hu C (2015) CloudFreq: Elastic energy-efficient bag-of-tasks scheduling in DVFS-enabled clouds. In: 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), IEEE, pp 585–592
30. Arunarani A, Manjula D, Sugumaran V (2019) Task scheduling techniques in cloud computing: a literature survey. Futur Gener Comput Syst 91:407–415
31. Saemi B, Sadeghilalimi M, Hosseinabadi AAR, Mouhoub M, Sadaoui (2021) A New Optimization Approach for Task Scheduling Problem Using Water Cycle Algorithm in Mobile Cloud Computing. In: 2021 IEEE Congress on Evolutionary Computation (CEC) IEEE, pp 530–539
32. Kashikolaei SMG, Hosseinabadi AAR, Saemi B, Shareh MB, Sangaiah AK, Bian G-B (2020) An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. J Supercomput 76(8):6302–6329
33. Shojafar M, Kardgar M, Hosseinabadi AAR, Shamshirband S, Abraham (2016) A: TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan. In: International Conference on Hybrid Intelligent System, Springer, https://doi.org/10.1155/2016/6675759
34. Gen M, Cheng R (1999) Genetic algorithms and engineering optimization. John Wiley Sons, New York
35. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat No 99TH8406), IEEE, pp 1470–1477
36. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE , pp 19421948
37. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
38. Chen X, Cheng L, Liu C, Liu Q, Liu J, Mao Y, Murphy J (2020) A woa-based optimization approach for task scheduling in cloud computing systems. IEEE Syst J 14(3):3117–3128
39. Alsaidy SA, Abbood AD, Sahib MA (2020) Heuristic initialization of PSO task scheduling algorithm in cloud computing. J King Saud Univ-Comput Inform Sci. https://doi.org/10.1155/2020/6675759
40. Mahmoodabadi M, Bagheri A, Nariman-Zadeh N, Jamali A (2012) A new optimization algorithm based on a combination of particle swarm optimization, convergence and divergence operators for single-objective and multi-objective problems. Eng Optim 44(10):1167–1186

41. Ramezani F, Lu J, Hussain F, (2013) Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. In: International Conference on Service-oriented Computing, Springer, pp 237–251
42. Javadpour A, Wang G, Rezaei S, Chend S (2018) Power curtailment in cloud environment utilising load balancing machine allocation. In: IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI); pp 1364–1370
43. Panda SK, Jana PK (2017) SLA-based task scheduling algorithms for heterogeneous multi-cloud environment. J Supercomput 73(6):2730–2762
44. https://code.google.com/p/hcsp-hc/source/browse/trunk/AE/ProblemInstances/HCSP/Braun_et_al/u_c_hihi.0?r=93
45. Javadpour A (2020) Providing a way to create balance between reliability and delays in sdn networks by using the appropriate placement of controllers. Wireless Pers Commun 110:1057–1071. https://doi.org/10.1007/s11277-019-06773-5

Authors and Affiliations

Poria Pirozmand[1] · Amir Javadpour[2,3] · Hamideh Nazarian[4] · Pedro Pinto[2] · Seyedsaeid Mirkamali[5] · Forough Ja'fari[6]

[1] School of Computer and Software, Dalian Neusoft University of Information, Dalian 116023, China

[2] ADiT-Lab, Electrotechnics and Telecommunications Department, Instituto Politécnico de Viana do Castelo, Porto, Portugal

[3] Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China

[4] Department of Management, Alzahra University, Tehran, Iran

[5] Department of Computer Engineering and IT, Payame Noor University (PNU), Tehran, Iran

[6] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran