Sellaperumal Parthasarathy* and Chinnasami Jothi Venkateswaran

# Deadline Constrained Task Scheduling Method Using a Combination of Center-Based Genetic Algorithm and Group Search Optimization

https://doi.org/10.1515/jisys-2017-0388
Received August 1, 2017; previously published online November 28, 2017.

**Abstract:** The present paper describes a hybrid group search optimization (GSO) and center-based genetic algorithm (CBGA)-based model for task scheduling in cloud computing. The proposed hybrid model combines the GSO, which has been successful in its application in task scheduling, with the use of the CBGA. The basic scheme of our approach is to utilize the benefits of both the GSO algorithm and CBGA excluding their disadvantages. In our work, we introduce the hybrid clouds, which are needed to determine which task to be outsourced and to what cloud provider. These choices ought to minimize the expense of running an allotment of the aggregate task on one or various public cloud providers while considering the application prerequisites, e.g. deadline constraints and data requirements. In the hybridization approach (HGSOCBGA), each dimension of a solution represents a task and the solution as a whole signifies all the task priorities. The vital issue is how to allocate the user tasks to exploit the profit of the infrastructure as a service (IaaS) provider while promising the quality of service (QoS). The generated solution proficiently assures the user-level QoS and improves the IaaS providers' credibility and economic benefit. The HGSOCBGA method also designs the hybridization process and suitable fitness function of the corresponding task. According to the evolved results, it has been found that our algorithm always outperforms the traditional algorithms.

**Keywords:** Hybrid GSO-CBGA, scheduling, cloud computing, quality of service, IaaS providers.

## 1 Introduction

Cloud computing has emerged as a developing model that is well geared to effectively exploit the manifold distributed resources that can be allotted to clients as per their needs. As a result, it leads to the economic and effective deployment of accessible and trouble-free management of gigantic computational issues. Further, it incredibly scales down the total outlay on several resources such as hardware and software [6], and easily facilitates the resources to be utilized for the purpose of leasing and releasing. Moreover, it boasts of a host of vantage points such as transparency of resources, flexibility, location independence, consistency, affordability, better accessibility of services, and so on [23]. For the purpose of enabling the related facilities, the tasks have to be planned suitably on the basis of resources in order to elicit the greatest accomplishment with the shortest time frame. Moreover, the cloud services are hosted on the infrastructure of the service providers themselves or those of the intermediary cloud infrastructure providers [19]. Basically, there are three types of services that are offered, such as the platform as a service (PaaS), infrastructure as a service (IaaS), and software as a service (SaaS) [4].

Further, the clouds may be broadly categorized into three types such as public, private, and hybrid [18]. When the cloud services are offered intended for the ordinary client on the basis of the pay-per-use pattern, it is labeled as the public cloud. When the institutions design their own unique applications and manage their own internal infrastructure, it is called the private cloud, as the access thereto is restricted to users within the institution [13]. The scheduling of the relative gigantic number of assignments has emerged as a vital problem

***Corresponding author: Sellaperumal Parthasarathy,** Faculty of MCA, Valliammai Engineering College, Anna University, Chennai, Tamil Nadu, India, e-mail: parthasarathys0667@gmail.com
**Chinnasami Jothi Venkateswaran:** Department of MCA, Presidency College, Chennai 5, Tamil Nadu, India

in the domain of cloud computing. In this regard, a lot of scheduling techniques are becoming popular, as illustrated in Refs. [11, 16, 22]. The corresponding techniques take into account a number of divergent features such as the cost matrix created by means of the credit of tasks to be allotted to a specific resource [22], quality of service (QoS)-based meta-scheduler and backfill strategy-based lightweight virtual machine scheduler for dispatching jobs [11], QoS needs [22], and heterogeneity of the cloud scenario and workloads [16].

Normally, the scheduling issue may involve two distinct categories, such as static and dynamic. As far as static scheduling is concerned, the attributes of an analogous program like the task processing periods, communication, data dependencies, and harmonization needs are identified well ahead of accomplishment [21]. In the case of dynamic scheduling, certain presumption regarding the identical program has to be spelt out prior to the performance, and thus the scheduling decisions have to made on the fly [15]. With the intention of effectively addressing scheduling issues, a novel cloud resources allotment structure has been introduced recently to facilitate the effective deployment of external clouds (ECs) so as to enable an IaaS cloud itself to become flexible [27]. In the relative configuration, an IaaS cloud is endowed with its own unique personal cloud and is capable of effectively outsourcing its functions to the parallel cloud providers known as the ECs, especially when its local resources are scarce. Further, each and every assignment carries with it a sharp cutoff date to complete the relative assignment in order for the resource allocation issue to be broadly deemed as a deadline constrained task scheduling (DCTS) one. An integer programming formulation of the DCTS issue is brilliantly brought to the limelight, with the intention of maximizing the benefit of the personal cloud on the concept of ensuring the QoS. The major contributions made in research for the task scheduling process are as follows:

- An approach named the HGSOCBGA is done for task scheduling, which has the advantages of easy realization and quick convergence, so that this scheduling approach is able to get an optimal or suboptimal solution with maximum profit than the individual group search optimization (GSO) and center-based genetic algorithm (CBGA) optimization algorithm.
- In the HGSOCBGA, the hybridization uses the process of best solution replacement ahead of the worst solution to improve the solution quality.

The organization of the paper is as follows. Section 2 presents a review of the literature. Section 3 describes the algorithm used in this research. Section 4 explains the solution framework and problem definition, and Section 5 describes the proposed task scheduling. Section 6 explains the experimental results, and Section 7 presents the conclusion part.

## 2 Literature Survey

Of late, the scheduling technique has began to play a critical part in modern applications, and notably, task scheduling has increasingly gained the attention of enthusiastic investigators, thanks to its extensive popularity, viability, and the copious expansion of the cloud computing system. The underlying motive behind cloud computing is furnishing effective accessibility to far-flung and geographically disseminated resources. In this regard, scheduling has emerged as one of them, as it indicates a group of policies to manage the order of job to be carried out with the help of a computer system. A proper scheduler has to invariably change its scheduling plans in phase with the ever-changing scenario and the category of the task (Table 1).

In this regard, Omara and Arafa [20] introduced genetic algorithms (GAs) for the purpose of successfully addressing the task scheduling problem. They effectively employed two genetic techniques to overcome the corresponding scheduling hurdles. In their novel modified GAs, they brilliantly brought in certain heuristic principles that were supplemented to enhance the efficiency of the accomplishment. With the intent to overcome the resultant problems, Abrishami and Naghibzadeh [1] elegantly launched the innovative deadline-constrained workflow scheduling in software as a service cloud. They brilliantly employed the partial critical Paths, which was very effective in drastically bringing down the expenditure related to the workflow accomplishment by appropriately satisfying a user-defined cutoff date. Further, for the purpose of cutting back the

**Table 1:** Parameters Used in Task Scheduling.

| Parameter | Variable description |
|---|---|
| $n$ | Number of cloud provider |
| $I$ | Number of VM type |
| $m$ | Number of application |
| $M_u$ | Price of the $u^{th}$ VM type in $CP_1^S$ |
| $C_{ku}$ | Cost of the $u^{th}$ VM type in $CP_k^S$ |
| $D_i$ | Deadline of the $i^{th}$ application |
| $R_i$ | Run time of each task in the $i^{th}$ application |
| $Q_{iu}$ | If $Q_{iu} = 1$, the $i^{th}$ application uses VM type $V_u^M$; otherwise, it does not use this type |
| $CPU_u$ | Number of CPUs for the $u^{th}$ VM type in $CP_1^S$ |
| $u^{th}$ | Size of memory for the $u^{th}$ VM type in $CP_1^S$ |
| Total_CPU | Total number of CPUs in $CP_1^S$ |
| Total_mem | Total size of memory in $CP_1^S$ |
| $S^S t_{il}$ | Integer decision variable |

cost of the processing considerably, Guo et al. [7] envisioned task scheduling with the help of the particle swarm optimization (PSO) algorithm, which is dependent on the small position value rule. In order to tackle the workflow issue, workflow scheduling meant for the cloud scenario dependent on the artificial bee colony algorithm was brought to the limelight by Kumar and Anand [14].

Moreover, Xu et al. [26] excellently envisioned an effective task scheduling scheme on heterogeneous computing systems by employing a multiple-priority queue GA. The novel technique integrated the GA technique to allocate a priority to each subtask while employing a heuristic-based earliest-finish time method in the hunt for a solution for the task-to-processor mapping. Singh and Singh [24] remarkably introduced the innovative score-based deadline constrained workflow scheduling algorithm, which was able to carry out the workflow within affordable expenses, simultaneously fulfilling the user-defined cutoff date stipulation. The novel technique elegantly employed the idea of a score that illustrated the capacities of hardware resources. Moreover, Agarwal and Jain [2] launched the optimal algorithm of task scheduling in the cloud computing environment. It represented a generalized priority technique for the effective performance of the assignments, and was further contrasted with the outcomes of the first come, first served and round robin scheduling. Similarly, Zuo et al. [27] excellently designed a self-adaptive learning PSO-based deadline constrained task scheduling for the hybrid IaaS cloud. The vital issue in the scheduling was concerned with the manner in which the assignments of the clients were to be allotted with the intention of taking the fullest benefit of the IaaS provider while assuring the QoS.

# 3 Algorithm Design

In this section, we first discuss the background of the GSO algorithm and CBGA. Then, the details of the proposed algorithm, HGSOCBGA, will be presented.

## 3.1 CBGA

The GA, kick-started by Holland in 1975 with a big bang [9], represents an iterative stochastic in which the natural evaluation is elegantly employed to model the search technique. These days, the novel technique is handed a red carpet welcome, thanks to its innate skill in successfully solving several optimization problems by emulating the genetic process of the biological organism [25]. As evident from the name, the

GA effectively imitates the evolutionary trend in nature to successfully address optimization issues. The straightforward GA effectively navigates through the three genetic functions of selection, crossover, and mutation. In the selection function, certain solutions from the populations are shortlisted as the parents. Further, in the crossover activity, the parents are subjected to the process of crossbreeding to generate the offspring. The final mutation function involves the task of orchestrating the offspring in accordance with the mutation regulations. In the novel technique, solutions are labeled as the "individuals" and the iterations of the technique are afforded the name "generations." In this regard, a lion's share of the genetic techniques invariably employs elitism, which indicates that a multitude of the best individuals are copied to the succeeding generation.

Of late, the GA has been afforded red carpet welcome in the domains of several scientific and engineering applications [10]. Nevertheless, in several engineering applications, a straightforward GA habitually encounters innate deficiencies like the untimely and sluggish convergence to the universal minimum. With an eye on alleviating the related defects, many variants of the GA have been popular. Jiaqing and Ling [12] deftly designed a CBGA in which a central chaotic mutation was devised to direct the evolutionary searching task by means of the data of the population center. Further, an innovative Cauchy preferential crossover operator was elegantly employed to investigate the search space. Their suggested technique was effectively performed to steer clear of the stiffness equivalence hassles of the small-aspect-ratio aircraft wing to the tapered beam. Enthused by the above-mentioned investigations, in the document, an earnest endeavor is made to launch the novel CBGA approach for the purpose of task optimization in the cloud computing structure for the purpose of considerable cost reduction.

Striking a quite divergent note from the time-tested search approaches, the CBGA effectively employs multiple search nodes concurrently. Each and every search node relates to one of the modern solutions and is characterized by a progression of symbols. The relative series is known as the chromosome, whereas the symbols constituting the series are named as the genes. Each chromosome is endowed with a linked value termed as the fitness value, which is effectively estimated by means of the objective function (fitness function) value $f(x)$. In the novel CBGA technique, only the excellent chromosomes enjoying superlative fitness values are found to survive longer and produce the offspring, thereby communicating their biological heredity to the successive generations. By developing the chromosome in a gradual manner, the solutions associated to the search nodes are augmented progressively. A set of chromosomes at a specified phase of the CBGA is known as the pop. The number of chromosomes (individuals) in a population is termed as the pop size. Moreover, the elitism size constitutes the number of fit individuals that are copied straight to the successive generation.

## 3.2 GSO Algorithm

The novel GSO technique was kick-started by He et al. [8]. It represents a population-based optimization approach that elegantly employs the producer-scrounger (PS) pattern and the animal scanning system. The PS technique devoted for devising the optimum searching technique had its humble origin, thanks to the motivation derived from the animal searching trend and the community living concept. It has habitually resulted in the espousal of two foraging techniques within the groups as follows: (i) one method is concerned with production and the hunt for food resources, and (ii) the other relates to the process of scrounging, or joining resources exposed by others. With the intention of not being trapped in the local minima, the GSO effectively utilizes the "ranger" foraging technique. The population of the GSO technique is known as a group and each individual in the population is labeled as a member. Altogether, there are three distinct types of members in the group, which are furnished as follows.

- **Producer:** The producer is entrusted with the task of generating appropriate stratagems and hunting for food sources. In each and every iteration, the member who has succeeded in locating the most gifted resource is shortlisted as the producer.
- **Scrounger:** The scrounger elegantly executes the scrounging techniques, joining resources exposed by others. Any member other than the producer in the group can be chosen as the scrounger.

– **Ranger:** The duties of the ranger include carrying out arbitrary walks and investigating the strategies for the arbitrarily disseminated resources. The members other than the producer and scroungers in a group are known as the rangers.

Recently, Couzin et al. [5] proposed that if the group is large, a small fraction of skilled persons is enough to direct the group with superlative precision. Hence, for the sake of precision and ease of evaluation, the PS technique is simplified based on the presumption that each searching bout contains a solitary producer and the residual members are either scroungers or rangers. Further, it is implicit that the producer, the scroungers, and the rangers do not exhibit any kind of divergence as regards the pertinent phenotypic attributes, enabling them to easily switch between the three diverse roles.

## 4 Solution Framework and Problem Definition

In this part, we offer a brief overview of the hybrid cloud architecture in order to additionally enhance the content of the research offered in this document. The overview of the hybrid cloud is given in Figure 1. Fundamentally, the hybrid cloud design is home to two types of clouds, such as the private cloud and the public cloud. The hybrid cloud in essence signifies a deft mixture of the public and private clouds. At times, it becomes essential for an entity to allocate a segment of its data as a public resource. Therefore, to preserve data secrecy, it appropriately seeks the aid of the hybrid cloud so that only limited resources are made public, without any damage to the secrecy of private resources that have to be kept intact as private. In the current investigation, with the ultimate objective of enabling the IaaS cloud itself adaptable, a novel cloud resource allocation structure is envisaged so as to facilitate the effective deployment of the ECs. Now, the IaaS cloud itself is endowed with its own unique private cloud and is competent to outsource its assignments to the parallel cloud providers known as the ECs, especially when it is resources-starved with regard to the local resources. In the relative structure, the ECs elegantly present a public interface for the generation and administration of the virtual machine (VM) instances within their proprietary infrastructure. From the outlook of an IaaS provider, the private clouds habitually represent its own resources while the ECs constitute the parallel

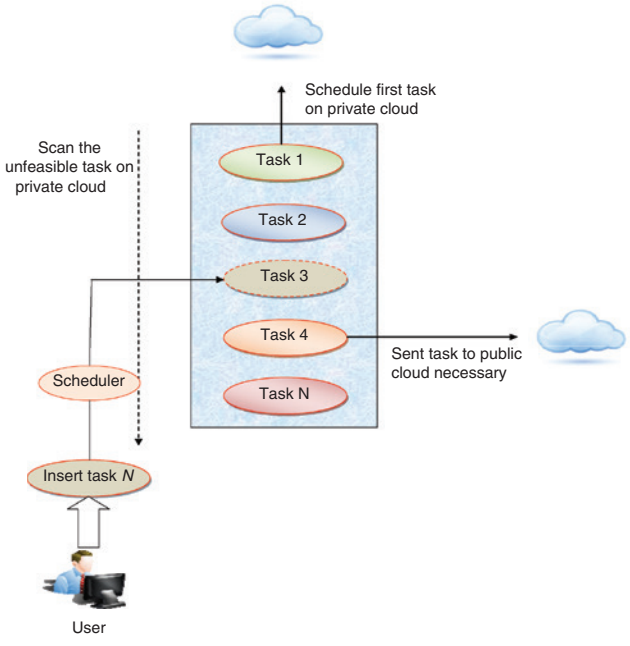


**Figure 1:** Basic Architecture of Hybrid Cloud.

clouds that are outsourced. In this regard, both the private clouds and ECs play their own unique role in this hybrid design.

In this regard, special mention has to be made regarding the vital contribution of scheduling in effectively addressing various functions within the hybrid cloud. In fact, scheduling deftly decides the nature of the task to be allocated in addition to the quantity of resources and computing components. It also prudently allots each and every task to the appropriate computing module. In case of availability of sufficient computational resources needed to perform the entire functions within the private cloud, it is not at all essential to accept on lease additional resources from the public cloud. However, in resource-crunched scenarios, making it difficult to accomplish the tasks within the deadline in the private cloud, the resources of the public cloud are availed on lease so that the essential assignments could be achieved well within the cutoff date. The resource allotment to the divergent assignments on the hybrid cloud is elegantly shown in Figure 1. It is clear from Figure 1 that the user assigns $n$ number of tasks to the scheduler. The scheduler module, in turn, gathers the relative scheduling data from the user and takes prudent decisions to allot each and every task to either the private cloud or one of the ECs existing in the public cloud, with the intention of taking the profit to hill level. In fact, the scheduling challenge has emerged as a deadline-inhibited task scheduling affair. In the novel technique, $m$ numbers of applications are employed, and each and every offered application consists of a host of analogous and self-governing chores, with a uniform and steady cutoff date common for all the constituent assignment tasks so that all the tasks are accomplished within the common deadline. Further, each and every task has to be performed in one VM instance type. Let $CP^S = \{CP_1^S, CP_2^S, ..., CP_n^s\}$ be a set of cloud providers. Here, $CP_1^S$ represents a private cloud and $CP_2^S, ...., CP_n^S$ corresponds to an EC. Further, $V^M = \{V_1^M, V_2^M, ..., V_I^M\}$ constitutes a set of VM types and $AaP = \{AaP_1, AaP_2, \ldots, AaP_m\}$ symbolizes a set of applications. Each application $AaP_i$ ($i \in \{1, 2, \ldots, m\}$) is assigned a fixed deadline $D_i$ and runtime $R_i$ and contains a task set $TK^i = \{T_1, T_2, ..., T_{iTK_i}\}$. Further, time is evidently characterized in the IP model by means of introducing time slots with a granularity of 1 h. The ultimate motive is to allocate $m$ applications to $CP_k^S$, where ($k \in \{1, 2, \ldots, n\}$) are to utilize the profit of $CP_1^S$. Further, each task has to be allotted to a single $CP_k^S$, $k \in \{1, 2, \ldots, n\}$. When a task $TK_i$ is initiated for performance, it is to be ensured that it is not interrupted and that its running slots are successive. In any time slot $S^S$ ($S^S \in \{1, 2, \ldots, S\}$), the resources deployed by the entire task accomplished in $CP_1^S$ should not exceed the total resources of $CP_1^S$ at any point of time from the perspective of $CP_1^S$, and all its ECs are endowed with never-ending sources. In task scheduling, each task has a strict deadline to meet, so that the resource allocation problem can be considered a DCTS one. To solve this problem nowadays [27], many optimization algorithms are used. In our work, we have used the hybrid GSO and CBGA-based scheduling approach in the hybrid cloud. Here, each submitted application contains a number of parallel and autonomous tasks and has a firm deadline of when all its consisting tasks must be completed. Each task needs to be implemented in one VM instance type only.

## 5 Proposed Task Scheduling Based on the GSO-CBGA Algorithm

In this section, we explain the proposed task scheduling using the hybrid GSO-CBGA algorithm. Task scheduling is one of the major problems in the cloud system. The main intention of the scheduling algorithm is to maximize the profit of the task, as well as minimize the execution time and cost of the task along with QoS requirements of the user. The step-by-step process is explained below.

**Step 1: Solution encoding**

In cloud computing, one of the vital ideas is to specify a solution for the task scheduling. Each application is home to a host of tasks, and hence all applications may be treated as a set of tasks as illustrated by $TK = \{T_1, T_2, ..., T_{TN}\}\left(TN = \sum_{i=1}^{w} T_i\right)$. Here, the solution is represented as a $TN(D = TN)$ dimension vector and each dimension (position) characterizes a task. The position with a bigger value indicates the fact that the task

**Table 2:** Solution Coding and Decoding.

| Dimension | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Position value | 7.16 | 7.34 | 6.52 | 4.32 | 5.28 |
| Priority | 2 | 1 | 3 | 5 | 4 |

specified by this position possesses a superior priority and has to be shortlisted first and foremost for the allotment of the scheduling process.

The ranked-order value rule [17] is employed to treat a particle $A_i = (a_{i1}, a_{i2}, \ldots, a_{iD})$ into a variant of tasks $TK = \{T_1, T_2, \ldots, T_D\}$ for the purpose of appraising the corresponding particle [3]. For example, let us consider a problem with five distinct tasks ($D = 5$); the $i^{\text{th}}$ particle is specified by $a_i = (7.16, 7.34, 6.52, 4.32, 5.28)$. The position $a_{i2}$ possesses the highest value, so that the task is represented by $a_{i2}$ and it is allotted a rank value one, as exhibited in Table 2. The next-in-line $a_{i1} = 6.14$ is allotted a rank value two. Similarly, the rank values of 3, 4, and 5 are allotted to $a_{i3}$, $a_{i5}$, and $a_{i4}$, respectively. Hence, a priority series of task is achieved as Priority = {2, 1, 3, 5, 4}. The search solution represents the observed task and its dimension is represented by $N \times M$.

$$A_i = \begin{bmatrix} a_{11} & a_{12},\ldots,a_{1D} \\ a_{21} & a_{22},\ldots,a_{2D} \\ a_{n1} & a_{n2},\ldots,a_{nD} \end{bmatrix}. \tag{1}$$

Here, $N$ indicates the number of the application existing in the investigation and $M$ represents the task.

**Step 2: Fitness calculation**

A fitness function is required when applying the hybrid GSA-CBGA to optimize the task permutation to maximize the profit of $CP_1^S$ and to work out the fitness value of each task in the application. Each task in set $T$ is allocated to one $CP_k^S$ ($k = 1, 2, \ldots, n$) according to the priority expressed by the code of a solution. We try to allocate each task to $CP_1^S$ because the cost of $CP_1^S$ is the lowest among all the clouds. If $CP_1^S$ has available resources to meet a task demand during its run time, then the task is allocated to $CP_1^S$; otherwise, the task is allocated to an EC with the minimal cost. The fitness value calculation formula is given in Eq. (4).

$$\text{Total\_income} = \sum_{l=1}^{D} \sum_{u=1}^{I} Q_{AaP(l)u} M_u R_{AaP(l)}. \tag{2}$$

$$EC_l = \underset{k \in \{2,\ldots,n\}}{\operatorname{argmin}} \left\{ \sum_{u=1}^{I} AaP(l)_u C_{ku} \right\}, \qquad l \in \{1, 2, \ldots, D\}. \tag{3}$$

$$\text{Fitness} = \text{total\_income} + \text{total\_cost}(T_l). \tag{4}$$

**Step 3: Divide the agent into two groups based on the hybrid probability *P***

One group uses the GSO to update their positions while the other uses the CBGA.

**Step 4: GSO stage**

We perform the GSO stage; here, we need three types of employees like the producer, scroungers, and rangers. These three employees make different operations to the population (task) and produce the optimal population.

(i) Producer operation

At first, the producer $A_p$ is located from the group member $A$ dependent on the best fitness function. The producer represents the best fitness value of $A_i$, which is represented by $A_P$.

The head angle of each and every individual is illustrated by means of Eq. (16) shown below.

$$\varphi_i^t = (\varphi_{i1}^t, ..., \varphi_{i(n-1)}^t) \in R^n. \tag{5}$$

The search direction of the member in accordance with the head angle is represented by the following equation:

$$U_i^k(\varphi_i^k) = (u_{i1}^k, ..., u_{in}^k) \in R^n. \tag{6}$$

The search direction is effectively evaluated from the head angle with the help of polar and Cartesian coordinate transformation, as illustrated by the following equations:

$$u_{i1}^k = \prod_{p=1}^{n-1} \cos(\varphi_{ip}^k). \tag{7}$$

$$u_{ij}^k = \sin(\varphi_{i(j-1)}^k) . \prod_{p=i}^{n-1} \cos(\varphi_{ip}^k). \tag{8}$$

$$u_{in}^k = \sin(\varphi_{i(n-1)}^k). \tag{9}$$

In the GSO technique, at the $k^{\text{th}}$ iteration, the producer $A_p$ exhibits the following conduct.

The producer performs the task of scanning at zero degree and thereafter continues the scan laterally by arbitrarily sampling the three points in the scanning domain:

– For the first point at zero degree, the scanning is as shown in Eq. (10) below:

$$A_z = A_p^k + r_1 l_{\max} H_p^k(\varphi_p^k). \tag{10}$$

– For the second point in the right-hand side in the hypercube, scanning is expressed in the following equation:

$$A_r = A_p^k + r_1 l_{\max} H_p^k\left(\varphi_p^k + r_2 \frac{\theta_{\max}}{2}\right). \tag{11}$$

– For the third point in the left-hand side in the hypercube, scanning is carried out by means of Eq. (12):

$$A_l = A_p^k + r_1 l_{\max} H_p^k\left(\varphi_p^k - r_2 \frac{\theta_{\max}}{2}\right). \tag{12}$$

Here, $r_1 \in R^1$ represents a generally disseminated arbitrary number with mean 0 and standard deviation 1, and $r_2 \in R^{n-1}$ indicates an evenly disseminated arbitrary sequence in the range (0, 1).

The maximum search angle $\theta_{\max}$ is furnished with the help of the following equation:

$$\theta_{\max} = \frac{\pi}{c^2}. \tag{13}$$

The constant $c$ is represented by Eq. (14), as follows:

$$c = \text{round}\left(\sqrt{n+1}\right). \tag{14}$$

Here, $n$ reveals the size of the search space:

$$\theta_{\max} = \frac{\pi}{n+1}. \tag{15}$$

The maximum distance $l_{\max}$ is effectively evaluated with the following equations:

$$l_{\max} = \| l_U - l_L \|. \tag{16}$$

$$l_{\max} = \sqrt{\sum_{i=1}^{n} (l_{Ui} - l_{Li})^2}. \tag{17}$$

Here, $l_{Ui}$ represents the upper bound for the $i^{\text{th}}$ dimension and $l_{Li}$ corresponds to the lower bound for the $i^{\text{th}}$ dimension. The best point with the best resource is effectively ascertained by means of Eqs. (7–9). In case the best point is endowed with a better resource than the present best position, then it moves to the new best point. Or else, it continues in its existing position and turns the producer head to the arbitrarily produced head angle direction, with the assistance of Eq. (18) given below:

$$\varphi^{k+1} = \varphi^k + r_2 \, \alpha_{\max}. \tag{18}$$

Here, $\alpha_{\max} \in R^1$ corresponds to the maximum turning angle.

If the producer fails to locate a superior area after $\alpha$ iterations, it turns its head back to zero degree as illustrated in Eq. (19) given below:

$$\varphi^{k+\alpha} = \varphi^k. \tag{19}$$

Here, $\alpha$ represents a constant.

(ii) Scrounger operation

At each iteration, in addition to the producer, there are a number of group members who are shortlisted as the scroungers. The common scrounging conduct in the GSO algorithm is the area copying trend. In $k^{\text{th}}$ iteration, the area copying conduct of the $i^{\text{th}}$ scrounger is formulated as a movement in the direction of the producer by means of Eq. (20) shown below:

$$A_i^{k+1} = A_i^k + r_3 \circ (A_p^k - A_i^k). \tag{20}$$

Here, $\circ$ represents the Hadamard product, which effectively evaluates the entry-wise product of the two vectors and $r_3 \in R^{n-1}$ uniform arbitrary sequence in the range (0, 1). The $i^{\text{th}}$ scrounger continues to be on the hunt for the optional opportunities to join. The corresponding trend is designed by turning the $i^{\text{th}}$ scrounger head to a new arbitrarily produced angle [Eq. (18)].

(iii) Ranger performances

The remaining group members that are isolated from their current positions are known as the rangers. They invariably carry out the searching tactics, which involve arbitrary walks and systematic searching strategies to effectively identify the resources. In this regard, the random walks emerge as the most ideal searching technique for the arbitrarily disseminated resources. It further generates an arbitrary head angle as illustrated in Eq. (18), and favors the arbitrary distance as the following equation:

$$l_i = a \cdot r_1 l_{\max}. \tag{21}$$

Further, the arbitrary walk to the new point is effectively exhibited in Eq. (22) shown below:

$$Y^{k+1} = Y_i^k + l_i H_i^k \varphi^{k+1}. \tag{22}$$

When all the above processes are finished, the fitness of the modernized solution is evaluated. The relative procedure is endlessly followed until the $l_i = k^{\text{th}}$ iteration so as to locate the best solution.

**Step 5: CBGA stage**

The CBGA effectively employs three distinct operators such as the selection, crossover, and mutation, which are elucidated below.

(i) Selection operation

Here, the roulette wheel selection approach is extensively employed in the CBGA selection procedure. It is capable of ensuring that the selection probability of every particle is in direct proportion to its fitness. In other words, if the fitness of a particle is superior, it becomes further eligible to get shortlisted.

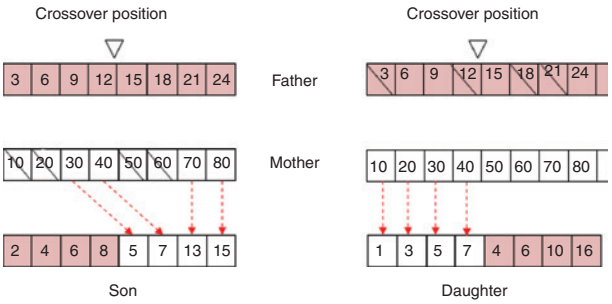


**Figure 2:** Single-Point Crossover Operator.

(ii) Crossover operation

When the selection function is over, it is followed by the execution of the crossover and mutation functions. The population of a center-based genetic technique takes its origin predominantly by means of the crossover and mutation functions. In the current investigation, the most leading genetic operator is the crossover, as it habitually modernizes the solutions almost always. Incidentally, a crossover represents the process of substituting certain genes in one of the parents by the matching genes of the other. In the task scheduling challenge, the crossover operator deftly blends two legitimate parents, whose subtasks are arranged topologically, to create two offspring, which also become legitimate. Figure 2 illustrates the single-point crossover model.

(iii) Mutation operation

When the crossover task is finished, for the purpose of increasing the skills to run away from the local optima, a central chaotic mutation is launched in accordance with the central data of the population. In this connection, the chaos represents a type of a non-linear dynamic that can be effectively employed for the universal optimization in view of its ergodicity and arbitrary nature. Further, a Loren system is elegantly utilized to generate the chaotic variables and thereafter infuse the chaotic variables into the chosen individuals so as to carry out the mutation function. The roadmap for the central chaotic mutation is drawn in the following section.

- Step 1: Choose the number of individuals from the population by means of the crossover function.
- Step 2: Create an eight-dimensional random vector $r_1$, each element of which is in the interval [0, 1], and thereafter generate new chaotic vectors with iteration $r_k = 4r_{k-1}(1 - r_{k-1})$, $k = 2, \ldots, CL$, where

$$CL = [\mu_r \times L \times T]. \tag{23}$$

$L$ represents the chaos length factor.

- Step 3: Map all the chaotic vectors to the search space, and chaotic individuals are achieved by means of Eq. (24):

$$A_i^{\text{new}} = PC_g + \xi(r_k - 0.5), \tag{24}$$

where $\xi$ is the factor to control the chaos scope, $r_k$ is a chaotic vector, and $PC_g$ is the population center.

The population center is calculated based on Eq. (25):

$$PC_G = \frac{2}{T(T+1)} \sum_{j=1}^{T} \text{pob}_j * \text{fit}(\text{pop}_j). \tag{25}$$

Here, $T$ represents the dimension of the population. It is clear from Eq. (25) that the population center represents the weighted center of the entire positions of the individuals, where the fitness value is employed as the weight. Obviously, the center may be changed at various generations and is employed to effectively direct the evolutionary process.

**Table 3:** Pseudo Code for the Proposed Hybrid Algorithm.

***Input:***
Parameter of GSO algorithm
Parameter of CBGA algorithm
Parameter of task scheduling
***Output:***
Scheduled task
***Assumption:***
Initial population $P_i$, fitness $FF_i$, crossover $C_r$, mutation $\mu_r$
***Initialization:***
Initialize the position of the population $A_i = (a_i^1, a_i^2, ..., a_i^d, ..., a_i^n)$,
head angle $\phi_i$ of all members
crossover rate $C_r$
***Start:***
Generate the initial population $A_i$, $i = 1, 2, \ldots, n$
Evaluate the fitness ($FF_i$) of the population
Set cycle to 1
***Repeat:***
***Stage 1: for GSO***
***While*** (the termination conditions are not met)
***for*** (each members *i* in the group)
*Choose producer*: Find the producer $A_p$ in the group;
*Perform producing*:
{
The producer will scan at zero degree and then scan laterally by randomly sampling three points in the scanning field using Eqs. (7)–(9).
Find the best point with the best resource (fitness value). If the best point has a best resource than its current position, then it will fly to this point. Otherwise, it will stay in its current position and turn its head to a new angle using Eq. (18).
If the producer cannot find a better area $\alpha$ iteration, it will turn its head back to zero degree using Eq. (19).
}
*Perform scrounging*: Randomly select 80% from the rest of the members to perform scrounging.
*Perform ranging:*
{
For the rest of the members, they will perform ranging:
To generate a random head angle using Eqs. (18) and (2)
To choose a random distance $l_i$ from the Gaussian distribution using Eq. (21)
Finally move to the new point using Eq. (22)
}
*Check feasibility*: Check weather each member of the group violates the constraints.
If it does, it will move back to the previous position to guarantee a feasible solution.
***end for***
Set $k = k + 1$;
***end while***
***Stage 1: for CBGA***
***while*** ($t \leq$ maximum iteration)
chose the roulette wheel selection strategy-based population selection
crossover operation is performed, which is given in Figure 2
population center is calculated based on Eq. (25)
new chromosome updation is based on mutation in Eq. (24)
*end while*
**Stage 3: Hybridization**
Combine the solutions GSO and CBGA
if ($GSO_{best} < CBGA_{best}$)
**Replace** $GSO_{best}$ **to** $CBGA_{best}$
*end if*
if ($GSO_{best} > CBGA_{best}$)
**Replace** $CBGA_{best}$ **to** $GSO_{best}$
*end if*
**Let** $g = g + 1$
***Output***
A scheduled task
***stop***

**Step 6: Hybridization**
While comparing the GSO and CBGA, if the GSO best fitness value ($GSO_{best}$) is less than the CBGA fitness value ($CBGA_{best}$), the best position of the GSO is replaced by that of the CBGA. Else, if the CBGA fitness value is less than the GSO fitness value, the position is replaced by the GSO solution.

**Step 7: Termination criteria**
The technique stops its operation only if the maximum number of iterations is attained and the solution that possesses the best fitness value is shortlisted and labeled as the best feature to the task scheduling. When the best fitness is achieved with the help of the HGSOCBGA technique, the chosen task is allotted for the cloud computing procedure. The HGSOCBGA-dependent task scheduling technique pseudo code is demonstrated in Table 3.

# 6 Results and Discussion

In this section, we discuss the results obtained from the proposed HGSOCBGA algorithm-based task scheduling technique (Figure 3). We have implemented our proposed task scheduling using Java (jdk 1.6) with cloud Sim tools, and a series of experiments have been performed in a 2-GHz dual-core PC machine with 4 GB main memory running a 64-bit version of Windows 2007. The experiment is carried out mainly based on the profit and run time.

The utilization rate of CPU or memory at the $s^{th}$ time slot for the private cloud is calculated by the following equation:

$$M(s) = \sum_{i=1}^{D} \text{Res}_i \, E_{is} \Big/ \text{Total-Res}; \quad s \in \{1, 2, \ldots, S\}, \tag{36}$$

where $\text{Res}_i$ is the number of CPU or the size of memory requested by task $T_i$ and Total-Res is the total CPU or memory in the private cloud.

The average utilization rate of CPU or memory is calculated by

$$AU = M(s)/S. \tag{37}$$

(i) Experimental design
In this, we utilize three types of problem instances and several experimental results to authenticate the efficiency of our approach. Each instance has different numbers of tasks, CPU, and memory utilization. The VM instance types and the personal cloud and the EC prices are placed based on our observation on public clouds, and are specified in Tables 4–8. Two kinds of resources, i.e. CPU and memory, are selected, as they are two of the most typical configurations in selecting a VM instance in the cloud. Now, we describe the different resources used in our experiments.

- Problem instance 1: In order to be able to compare the output of the hybrid task scheduling algorithm with the optimal solution, the scale of the problem has to be relatively small. Here, in instance 1, the number of application is 8, the number of task is 20, and memory utilization is 40 GB. The VM instance type requested by each application is arbitrarily chosen from the above three VM types. The deadline of each application is a consistently distributed random integer between 1 and 5 h in order to limit the search space. To make certain that the deadline of each application is longer than its runtime, the runtime is selected as a consistently allocated integer between 1 h and its deadline.
- Problem instances 2 and 3: In problem instance 2, the number of applications is 5, the number of tasks is 512, and the memory utilization is 1024 GB. Similarly, problem instance 3 consists of 10 applications, 512 tasks, and memory utilization of 1024 GB. These two instances are large in size and are used for handling large-sized problems. Instances 2 and 3 are the most vital tasks and resources compared to instance 1.

(ii) Performance evaluation

The basic idea of our research is to schedule the task using the HGSOCBGA optimization algorithm. Here, the performance of the proposed approach is mainly evaluated using the maximum profit and average run time. Figures 4–6 show the performance of the proposed approach.

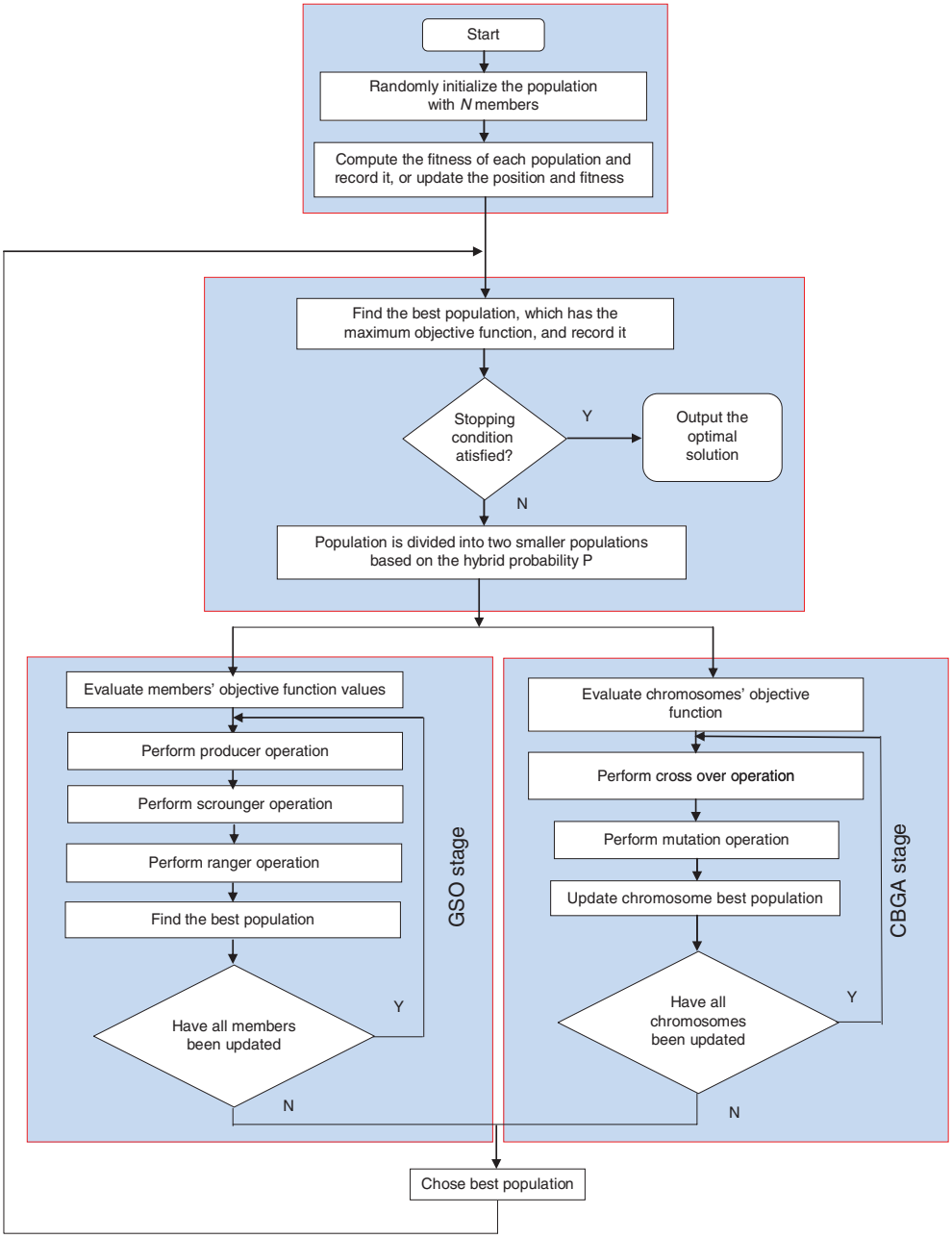


**Figure 3:** Overall Diagram of Our HGSOCBGA.

**Table 4:** VM Instance Types.

| Name | CPU | Memory |
|---|---|---|
| Small | 1 | 1.7 |
| Large | 4 | 7.5 |
| X large | 8 | 15 |

**Table 5:** Private Cloud Cost and Price.

| | Cost | Price |
|---|---|---|
| Small | 0.03 | 0.08 |
| Large | 0.12 | 0.32 |
| X large | 0.24 | 0.64 |

**Table 6:** EC Price.

| EC | Small | Large | X large |
|---|---|---|---|
| A | 0.085 | 0.34 | 0.68 |
| B | 0.070 | 0.30 | 0.70 |
| C | 0.100 | 0.40 | 0.72 |

**Table 7:** Parameters of Problem Instance 1.

| Application | | Cloud resources | |
|---|---|---|---|
| Parameters | Values (integer) | Resources | Number |
| Number of tasks | ~unit[1, 5] | CPU | 20 |
| VM instance type | ~unit[1, 3] | Memory | 40 GB |
| Deadline (h) | ~unit[1, 5] | | |
| Runtime (h) | ~unit[1, Deadline] | | |

**Table 8:** Parameters of Problem Instances 2 and 3.

| Application | | Cloud resources | |
|---|---|---|---|
| Parameters | Values (integer) | Resources | Number |
| Number of tasks | ~unit[1, 50] | CPU | 512 |
| VM instance type | ~unit[1, 3] | Memory | 1024 GB |
| Deadline (h) | ~unit[1, 168] | | |
| Runtime (h) | ~unit[1, Deadline] | | |

## 6.1 Discussion

The main intention of our work is task scheduling based on the optimization algorithm. Here, we have used the hybridization approach (HGSOCBGA) for the scheduling. The GSO approach is one of the optimization algorithms used to select the optimal or suboptimal solution, which is stimulated by animal behavior, especially animal searching behavior. In addition, the CBGA is the optimization algorithm that searches the best solution at the multi-dimensional search space according to the "best lives" principle. By hybridizing these two optimization algorithms, we assume that the optimization performance will be increased simultaneously, helping improve the profit and run time. By our assumptions, this result section shows that the proposed algorithm of the HGSOCBGA has achieved better performance than the individual algorithms. Table 8 shows the performance of the maximum profit and average time for problem instances 1, 2, and 3. From the table, in problem instance 1, we clearly understand that our proposed approach HGSOCBGA achieves the maximum profit of 8.91, which is 7.64 for using the GA-based task scheduling, 8.01 for using the GSO-based task scheduling, 8.41 for using the SLPSO-based task scheduling, and 8.25 for using the GSO-GA-based task

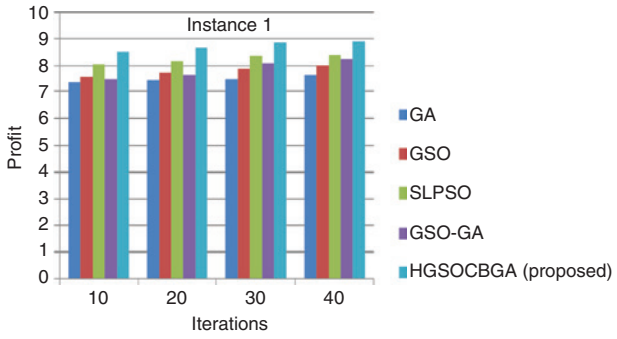


**Figure 4:** Performance of Profit Plot for Proposed against Existing Algorithm for Instance 1.

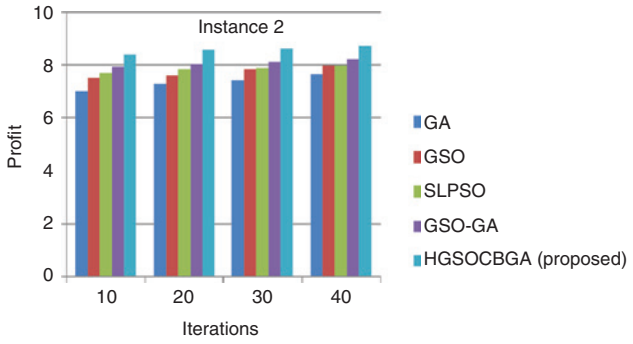


**Figure 5:** Performance of Profit Plot for Proposed against Existing Algorithm for Instance 2.

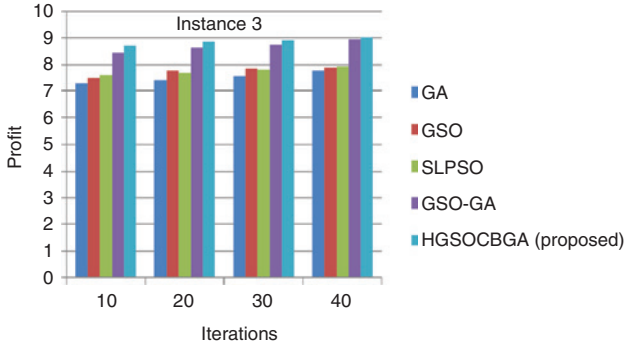


**Figure 6:** Performance of Profit Plot for Proposed against Existing Algorithm for Instance 3.

scheduling. When we use problem instance 2, we achieve the maximum profit of 8.74, which is high compared to the profit of existing approaches. Similarly, using instance 3, we also obtain the maximum profit of 9.04 by using our proposed approach (Table 9).

Table 10 shows the performance of the average memory utilization rate for the proposed approach. Here, we obtain the maximum memory utilization rate of 6,584,384 by using instance 1, 6,176,474 by using instance 2, and 8,154,651 by using instance 3. Table 11 shows the performance of the resource utilization ratio. From the table, we understand that our proposed approach achieves the maximum resource utilization ratio of 92 for instance 1, 91 for instance 2, and 93 for instance 3. Similarly, Table 12 shows the comparison of the CPU utilization rate. Figures 4–6 show the performance of the three problem instances based on the profit. The horizontal axis represents the number of iterations and the vertical axis represents the profit value. The suggested

**Table 9:** Parameters of Problem Instances 2 and 3.

| Algorithms | Problem instance 1 | | Problem instance 2 | | Problem instance 3 | |
|---|---|---|---|---|---|---|
| | Maximum profit | Avg. run time (ms) | Maximum profit | Avg. run time (ms) | Maximum profit | Avg. run time (ms) |
| GA | 7.64 | 8126 | 7.68 | 7498 | 7.79 | 9351 |
| GSO | 8.01 | 8624 | 7.98 | 8123 | 7.91 | 8612 |
| SLPSO | 8.41 | 9471 | 8.01 | 8674 | 7.93 | 7641 |
| GSO-GA | 8.25 | 9354 | 8.27 | 9872 | 8.94 | 9541 |
| CBGA-GSO (proposed) | 8.91 | 9547 | 8.74 | 9321 | 9.04 | 9647 |

**Table 10:** Comparison for Memory Utilization Rate.

| Average memory utilization rate | | | |
|---|---|---|---|
| **Algorithm** | **Instance 1** | **Instance 2** | **Instance 3** |
| GA | 6,146,141 | 6,587,414 | 6,971,789 |
| GSO | 5,951,165 | 6,761,114 | 6,951,141 |
| SLPSO | 4,816,167 | 7,594,141 | 7,941,561 |
| GSO-GA | 7,164,141 | 7,646,416 | 8,046,464 |
| CBGA-GSO (proposed) | 6,584,384 | 6,176,474 | 8,154,651 |

**Table 11:** Comparison for Resource Utilization Rate.

| Resource utilization ratio | | | |
|---|---|---|---|
| **Algorithm** | **Instance 1** | **Instance 2** | **Instance 3** |
| GA | 76 | 79 | 78 |
| GSO | 81 | 82 | 78 |
| SLPSO | 79 | 82 | 84 |
| GSO-GA | 74 | 80 | 85 |
| CBGA-GSO (proposed) | 92 | 91 | 93 |

**Table 12:** Comparison for CPU Utilization Rate.

| Average CPU utilization rate | | | |
|---|---|---|---|
| **Algorithm** | **Instance 2** | **Instance 2** | **Instance 3** |
| GA | 0.445632 | 0.45335 | 0.46332 |
| GSO | 0.55621 | 0.54224 | 0.613325 |
| SLPSO | 0.71245 | 0.68441 | 0.72335 |
| GSO-GA | 0.913254 | 0.84653 | 0.79555 |
| CBGA-GSO (proposed) | 0.953254 | 0.85775 | 0.807566 |

approach attains the maximum profit in 40 assessments for the small-size instance, which is high compared with the other algorithms like FA, GSA, SLGSA, SLPSO, and HSLGSAFA. The standard profit is acquired in 40 runs of the FA, GSA, SLGSA, SLPSO, and HSLGSAFA, and their average runtimes are specified in Table 9. From the results, we clearly understand that our proposed hybrid approach yields better performance compared to existing approaches.

# 7 Conclusions

In this paper, we have described a method based on a technique to solve the task scheduling problem using the hybridization approach. The main problem of task scheduling is how to assign the user task to maximize the profit of the IaaS provider while guaranteeing the QoS. By hybridizing the two optimization algorithms like the GSO and CBGA, this approach competently attains a high quality of scheduling. In the experimentations, we have carried out three types of problem instances having various numbers of tasks and applications. Here, our novel technique achieves the maximum profit of 9.04, which is better compared to that of the existing approaches.

# Bibliography


[1] S. Abrishami and M. Naghibzadeh, Deadline-constrained workflow scheduling in software as a service cloud, *Sci. Iran.* **19** (2012), 680–689.

[2] A. Agarwal and S. Jain, Efficient optimal algorithm of task scheduling in cloud computing environment, *Int. J. Comput. Trends Technol.* **9** (2014).

[3] J. C. Bean, Genetic algorithms and random keys for sequencing and optimization, *ORSA J. Comput.* **6** (1994), 154–160.

[4] S. Bhardwaj, L. Jain and S. Jain, Cloud computing: a study of infrastructure as a service (IaaS), *Int. J. Eng. Inf. Technol.* **2** (2010), 60–63.

[5] I. D. Couzin, J. Krause, N. R. Franks and S. A. Levin, Effective leadership and decision-making in animal groups on the move, *Nature* **434** (2005), 513–516.

[6] J. Geelan, Twenty-one experts define cloud computing virtualization, *Electron. Mag.* (2009).

[7] L. Guo, S. Zhao, S. Shen and C. Jiang, Task scheduling optimization in cloud computing based on heuristic algorithm, *J. Netw.* **7** (2012), 547–553.

[8] S. He, Q. H. Wu and J. R. Saunders, A group search optimizer for neural network training, *Lecture Notes Comput. Sci.* **3982** (2006), 934–943.

[9] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[10] T. H. Hsu, T. N. Tsai and P. L. Chiang, Selection of the optimum promotion mix by integrating a fuzzy linguistic decision model with genetic algorithms, *Inform. Sci.* **179** (2009), 41–52.

[11] R. Jeyarani, R. Ram, R. Vasanth and N. Nagaveni, Design and implementation of an efficient two-level scheduler for cloud computing environment, in: *Advances in Recent Technologies in Communication and Computing, ARTCom'09*. International Conference on, Koyyayam, Kerala, India, pp. 884–886, IEEE, 2009.

[12] Z. Jiaqing and W. Ling, Center based genetic algorithm and its application to the stiffness equivalence of the aircraft wing, *Expert Syst. Appl.* **38** (2011), 6254–6261.

[13] R. Kumar, Data security in cloud computing and cost analysis: review, *Int. J. Tech. Res. Appl.* **1** (2013), 40–46.

[14] P. Kumar and S. Anand, An approach to optimize workflow scheduling for cloud computing environment, *J. Theor. Appl. Inform. Technol.* **57** (2013), 617–623.

[15] Y. Kwok and I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, *ACM Comput. Survey* **31** (1999), 406–471.

[16] G. Lee, B.-G. Chunz and R. H. Katzy, Heterogeneity-aware resource allocation and scheduling in the cloud, in: *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, University of California, Berkeley, CA, USA, 2011.

[17] B. Liu, L. Wang and Y. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **37** (2007), 985–997.

[18] J. C. Mace, A. V. Moorsel and P. Watson, The case for dynamic security solutions in public cloud workflow deployments, in: *Proceeding of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops*, Hong Kong, China, 2011.

[19] L. Mei, W. K. Chan and T. H. Tse, A tale of clouds: paradigm comparisons and some thoughts on research issues, *IEEE Asia-Pacific Services Computing Conference (APSCC)*, Yilan, Taiwan, pp. 464–469, 2008.

[20] F. A. Omara and M. M. Arafa, Genetic algorithms for task scheduling problem, *J. Parallel Distrib. Comput.* **70** (2010), 13–22.

[21] M. A. Palis, J. C. Liou, S. Rajasekaran, S. Shende and S. S. L. Wei, Online scheduling of dynamic trees, *Parallel Process. Lett.* **5** (1995), 635–646.

[22] M. L. M. Peixoto, M. J. Santana, J. C. Estrella, T. C. Tavares, B. T. Kuehne and R. H. C. Santana, *A Metascheduler Architecture to Provide QoS on the Cloud Computing*, IEEE, Doha, Qatar, 2010.

[23] B. P. Rimal, E. Choi and I. Lumb, A taxonomy and survey of cloud computing systems, in: *Proceedings of the 5th IEEE International Joint Conference of INC, IMS and IDC*, Seoul, South Korea, pp. 44–51, 2009.

[24] R. Singh and S. Singh, Score based deadline constrained workflow scheduling algorithm for cloud systems, *Int. J. Cloud Comput. Services Architect.* **3** (2013).
[25] S. Song, K. Hwang and Y.-K. Kwok, Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling, *IEEE Trans. Comput.* **55** (2006), 703–719.
[26] Y. Xu, K. Li, J. Hu and K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, *Inform. Sci.* **270** (2014), 255–287.
[27] X. Zuo, G. Zhang and W. Tan, Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud, *IEEE Trans. Autom. Sci. Eng.* **11** (2014), 564–573.