Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 6516482, 9 pages
https://doi.org/10.1155/2023/6516482

*Research Article*

# A Hybrid Gravitational Emulation Local Search-Based Algorithm for Task Scheduling in Cloud Computing

**S. Phani Praveen**,[1] **Hesam Ghasempoor**,[2] **Negar Shahabi**,[3] **and Fatemeh Izanloo**[4]

[1]*Department of CSE, PVPSIT, Vijayawada, Andhra Pradesh, India*
[2]*Department of IT Management, Hadaf Institute of Higher Education, Sari, Iran*
[3]*Department of Computer Science, Islamic Azad University Babol Branch, Babol, Iran*
[4]*Department of Computer Science, Rouzbahan Institute of Higher Education, Sari, Iran*

Correspondence should be addressed to Hesam Ghasempoor; h.ghasempoor2020@gmail.com

Received 13 July 2022; Revised 7 November 2022; Accepted 25 November 2022; Published 4 February 2023

Academic Editor: Ardashir Mohammadzadeh



The flexibility of cloud computing to provide a dynamic and adaptable infrastructure in the context of information technology and service quality has made it one of the most challenging issues in the computer industry. Task scheduling is a major challenge in cloud computing. Scheduling tasks so that they may be processed by the most effective cloud network resources has been identified as a critical challenge for maximizing cloud computing's performance. Due to the complexity of the issue and the size of the search space, random search techniques are often used to find a solution. Several algorithms have been offered as possible solutions to this issue. In this study, we employ a combination of the genetic algorithm (GA) and the gravitational emulation local search (GELS) algorithm to overcome the task scheduling issue in cloud computing. GA and the particle swarm optimization (PSO) algorithms are compared to the suggested algorithm to demonstrate its efficacy. The suggested algorithm outperforms the GA and PSO, as shown by the experiments.

## 1. Introduction

Cloud computing, established in late 2007, has several capabilities such as providing flexible and dynamic infrastructure in the context of information technology, quality assurance criteria in computing environments, and configurable software services [1–3]. Because of the novelty and growth of cloud computing, there is no exact and standard description that specifies all of the aspects and qualities of technology, and each of the current definitions considers certain elements of this technology [4, 5]. Some of these definitions are as follows:

Cloud computing is a payment model for the ability to use customizable computing resources across shared networks such as networks, servers, storage space, applications, services, and so on. The payment model is based on the number of existing requests and the level of access to the network, which can be provided quickly and with the least amount of management and intervention from the service provider users [6, 7].

Cloud computing is a model for distributing data and computing over a large network of nodes. User PCs, data centers, and cloud services are examples of nodes. In reality, a network of nodes is a term used to describe the cloud [8, 9]. A cloud system is a form of parallel and distributed system made up of a collection of virtual machines (VMs) that supply computing resources in accordance with a service-level agreement (SLA) and via an agreement between the service provider and clients [10–12].

Shortly, task scheduling became an important issue in the cloud computing environment. The task scheduling algorithm is the method by which tasks are assigned to central data sources. Due to the variety of scheduling objectives, no complete scheduling algorithm with a precise solution is provided. An ideal scheduler is implemented

through agreement or, depending on the application, a combination of scheduling algorithms [13].

Depending on the algorithm used, a scheduling problem can be solved in seconds, hours, or even years. The time required to execute an algorithm is used to assess its performance. The algorithm's execution time is proportional to its input as a function of complexity time. Several time complexity algorithms are presented in [14]. A problem with polynomial time complexity is controllable, practical, and fast enough to run on computing machines. A complexity class is defined in computational complexity theory as a set of problems that have the same complexity due to a definite source [15, 16].

We intend to solve the problem of task scheduling in cloud computing with a new combination based on the genetic algorithm (GA) [17] and the gravitational emulation local search (GELS) algorithm [18, 19] in this article.

The novelties of this study are as follows:

(i) Considering new and different constraints to simulate the problem

(ii) It has maximum CPU utilization and execution time

(iii) Minimizing makespan and maximizing resource utilization in cloud computing

The structure of the paper is organized as follows: in Section 2, the related work is described. In Section 3, the problem statement is explained. The proposed algorithm is fully described in Section 4. The simulation results and conclusions are given in Sections 5 and 6, respectively.

## 2. Related Work

Many techniques have been presented for scheduling tasks in the cloud. This section has covered some ground in exploring these algorithms. Cloud task mapping is discussed in [20], where a novel approach is offered. The suggested technique has two primary phases: first, a modified honeycomb algorithm is used to prioritize jobs, and then another algorithm is used to organize tasks according to their relative importance. The algorithm for scheduling tasks takes into account the time and money needed to acquire the necessary resources. As a result of this algorithm's improvements, the ratio between the cost of acquiring resources and the cost of communicating effectively to get things done has decreased dramatically. This algorithm does not take into account the efficiency of its resources or the time it takes to run.

Liu et al. describe a cloud computing scheduling technique in [21]. Workflow compression is used in the proposed technique to minimize runtime and cost depending on user input information, which the user provides into the system at any time. Resource optimization is not taken into account in this method.

Scheduled tasks using the parameters of EFT, which represents the closest finish time of a task in a resource, are presented in [22]. ERT, which indicates the remaining time to perform a task, and EST, which provides the closest start time for a work, are considered in this research. Experiment findings show that, in addition to improving execution time, this approach may help raise or reduce the amount of resources consumed during execution. Cost considerations and resource optimization are not taken into account in this approach. The research presented by Wu et al. [23] involves both service-level and task-level scheduling. Service-level scheduling is accomplished by allocating work to services, while task-level scheduling is accomplished by optimally allocating jobs to VMs in cloud data centers. This method focuses on resource optimization while ignoring implementation time and cost.

In [24], an ant nest algorithm optimization is used to propose a service flow scheduling model in cloud computing based on several service quality variables. This method focuses on reducing the program's execution time while ignoring requirements for efficient resource and cost allocation. The key step of the algorithm in [25] is the selection of tasks and resources from the public cloud, as well as the development of a hybrid cloud, while the scheduler determines which processes will use public cloud resources to minimize runtime. The new timetable places a premium on determining performance and implementation costs. The source optimization criteria are ignored in this model. The major goal of task coordination and allocating work is provided in [26–28] to the service in line with the executive requirements in order to perform the correct thing and observe their priority. This approach assigns tasks to multiple levels and maps each level of task to resources with the capability of processing them.

Chiang et al. [29] proposed a BCSV scheduling algorithm to improve task scheduling in cloud computing. The main idea of the BCSV algorithm is to use the smallest suffrage value (SSV), the largest suffrage value (LSV), and the criteria suffrage value (CSV) as the scheduling factors to increase the job dispatch performance. According to the test results, the proposed BCSV algorithm can achieve better load balance and lifetime than the existing algorithms under the heterogeneous network environment of HiHi, HiLo, LoHi, and LoLo. In other words, the proposed BCSV algorithm can obtain better results in task scheduling while considering the load balancing problem.

Liu [10] has designed an adaptive task scheduling algorithm for cloud computing based on the ant colony algorithm (ACO) in order to solve the shortcomings of the ACO in solving scheduling problems. Based on the polymorphic ACO, the convergence speed of the algorithm is improved and effectively prevents the emergence of local optimal solutions. The objective of the improved algorithm is to solve a distribution scheme with shorter execution time, lower cost, and a balanced load rate based on tasks submitted by users. Experimental data showed that the improved adaptive ant colony algorithm (IAACO) was able to quickly find the optimal solution for the cloud computing resource scheduling problem, shorten the task completion time, reduce the execution cost, and maintain the load balance of the entire cloud system center.

Manikandan et al. [14] have proposed a new hybrid whale optimization algorithm (WOA) based on the MBA

algorithm in order to solve the multiobjective task scheduling in cloud computing. In the combined WOA-based MBA algorithm, multiobjective behavior minimizes lifetime by maximizing resource utilization. The output of the random double adaptive whale optimization algorithm (RDWOA) is increased by using the jump operator of the Bees algorithm. Performance evaluations are performed and compared with other algorithms using the CloudSim toolkit platform for various measures such as completion, time, and computational cost. The results are analyzed for performance measures such as build time, execution time, resource usage, and computational cost. The proposed algorithm performs better than other algorithms such as the improved whale optimization algorithm (IWC), modified ant lion optimization (MALO), bat algorithm with ant-bee colony optimization algorithm (BA-ABC), and modified genetic algorithm combined with greedy strategy (MGGS).

Guo [15] proposed a cloud computing multiobjective task scheduling optimization based on the fuzzy self-defense algorithm, presented the shortest time, degree of resource load balance, and multiobjective task completion as the goal of cloud computing multiobjective task scheduling, and established a mathematical model to measure the effect of multiobjective task scheduling. It showed that the cloud computing multiobjective task scheduling optimization method based on the fuzzy self-defense algorithm can improve the performance of the maximum multiobjective completion time, the deadline violation rate, and the use of VM resources.

Manikandan et al. [30] proposed a solution for the main problem in computing, which is scheduling and resource allocation: a solution with hybrid algorithms of fuzzy C-means clustering to use black widow optimization for task scheduling and fish swarm optimization (FSO) for efficient allocation of resources to reduce cost, energy, and use the resources provided. There was no proper method or technique to improve task scheduling and resource allocation. Previous methods used VM instances for scheduling. The main drawback of using VM instances was that it took a lot of setup time and all the resources to do the work.

Shukri et al. [31] have proposed an enhanced version of the multiverse optimizer (EMVO) as a superior task scheduler in this field in order to solve the task scheduling problem and minimize the execution time and cost. The main multiverse optimizer (MVO) and particle swarm optimization (PSO) algorithms are compared in cloud environments. The results show that EMVO significantly outperforms both MVO and PSO algorithms in terms of achieving minimum construction time and increasing resource utilization.

Arzoo [32] presented the ant particle swarm genetic algorithm (APSGA) which is a combination of PSO-ACO-GA to solve the task scheduling problem. PSO is inspired by the movement of the bird, ACO is based on the behavior of ants, and GA works on the complementary process. For this algorithm, there are tasks such as PSO, the firefly algorithm (FA), ACO, and GA. The proposed APSGA algorithm decreases PSO, ACO, and GA by 27.1%, 19.45%, and 21.24%, respectively to achieves maximum CPU utilization and runtime.

Khan and Santhosh [33] have presented a work scheduling method based on a hybrid optimization algorithm that has improved parameters such as waiting time, total production time, execution time, efficiency, and utilization. In terms of performance, this scheduling method is superior to the scheduling algorithms based on the optimization of PSO and ACO.

Mangalampalli et al. [34] introduced the cat swarm optimization algorithm (CSOA), which addresses the parameters makespan, migration time, energy consumption, and total power cost at data centers. The implementation was performed using the CloudSim simulator, and the input to the algorithm was randomly generated from CloudSim for the total energy cost; the parallel workloads of HPC2N and NASA were used as the input of the algorithm. This proposed algorithm was able to reduce energy consumption and minimize immigration by a significant percentage.

Amer et al. [35] presented a modified Harris Hawks optimizer (HHO) called the elite learning Harris Hawks optimizer (ELHHO), in relation to multiobjective scheduling. This paper showed the performance of ELHHO compared to conventional HHO and service quality in terms of minimizing schedule length, execution cost, and maximizing resource utilization.

## 3. Problem Statement

Scheduling tasks in a cloud computing environment is a major concern. Any user may access data stored in the cloud. Hundreds of digital tools are at your disposal for any activity. It is not feasible for the user to delegate tasks to virtual resources. Cloud computing is used so that service providers may reduce the money they spend on resource use while increasing the money they make supporting client applications. The scheduling system manages the numerous cloud-based jobs to boost completion times, resource productivity, and ultimately, computing power. Clouds may be seen as digitally equivalent to tree trunks and branches. That is why [1], we should all care about how jobs are scheduled across heterogeneous physical devices.

Cloud computing involves $n$ dependent tasks that must be processed by $m$ virtual dependent sources. $T = (t_1, t_2, \ldots, t_3)$ is a set of work that may depend on the work $t_j$ for which the cost of communication is considered, and also, $D = (d_1, d_2, \ldots, d_n)$ is the total number of resources available in the case of communication and the source of the cost of communication. The goal of scheduling in a cloud computing environment is to minimize total cost. The following are some of the constraints of the task scheduling issue in the context of cloud computing that are discussed in this work:

(i) Each case runs on only one resource

(ii) All resources are available in zero time

(iii) Maximum number of resources with minimum power consumption is used to schedule tasks

(iv) Scheduling is scalable, meaning it responds to any size of task input in the system for timing

## 4. The Proposed Method

The GA excels at exploring the problem space, but it struggles with stability and local search. In this work, we employ the combination of GA and GELS to address the job scheduling problem in cloud computing. The suggested algorithm's goal is to integrate the GA's general search capabilities with those of a more localized GELS. In the proposed algorithm (GAGELS), the goal is to reduce the total cost of performing all tasks. To better understand the problem, the scheduling of six tasks across five sources is shown in the form of a visual diagram in Figures 1 and 2.

Dependent tasks are represented as a graph whose vertices represent the tasks and whose edges show the cost of communication between related tasks. The cost of communication is received as an $N \times N$ matrix as input.

The general steps of the proposed algorithm are shown in Figure 3. As can be seen in this flowchart, the GA first tries to produce better chromosomes by genetic operators, and then these chromosomes are improved by the GELS and returned to the population. In fact, in this algorithm, the purpose of using GELS is to locally improve the solutions produced by the GA.

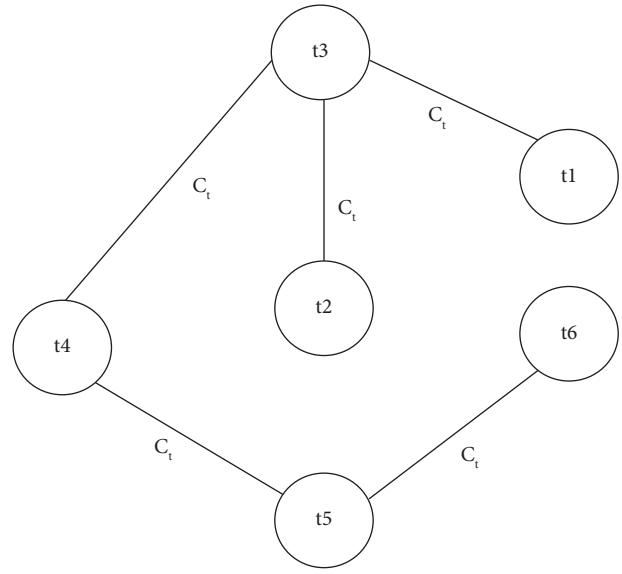


Figure 1: Illustration of the relationship of tasks as a weighted graph. $t$ represents a task and $C$ is the cost of communication between related tasks.

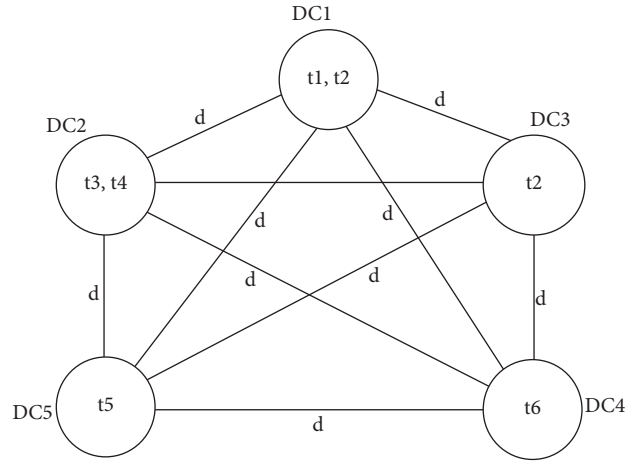


Figure 2: Display of the relationship of resources as a weighted graph. In this figure, $t$ is a work and $DC$ is the number of data centers in a node.

*4.1. Chromosome Structure.* Each chromosome in the GAGELS represents a possible solution to the problem and is a vector = $(d_1, d_2, \ldots, d_n)$; it is shown that $n$ is equal to the total number of tasks. The values of $d$ are represented as $TR$, where $T$ represents the task number and $R$ represents the source number. Figure 4 shows an example of a chromosome with 7 genes. For example, in this figure, the second gene is 23, which means that the second task is processed at source three.

*4.2. Fitness Function.* The goal of task scheduling in a cloud computing environment is to minimize the total cost, and this goal will be achieved if the restrictions are met. Costs include migration costs, operating costs, and communication costs. These restrictions are obtained by satisfying them. The competence of chromosomes is based on the cost of servicing tasks. In the proposed algorithm, equation (1) is used to obtain the competence of chromosomes:

$$F(x) = C_m + C_E + C_c. \tag{1}$$

In this equation, $C_m$, $C_E$, and $C_C$ are defined as migration cost, operation cost, and communication cost in a schedule, respectively.

*4.3. Parental Selection.* In the GAGELS, the ranking method is used to select the parent. Selection by the ranking method is such that all chromosomes are ranked according to equation (2) based on fitness:

$$Chr_{i_{1\le i\le n}} = (\text{Max} - \text{Fit}_i) + 1, \tag{2}$$

where $Chr_i$ is the chromosome rank $i$, Max is the greatest competency, that is, the worst competency, and $\text{Fit}_i$ is the competence of the $i$ chromosome. The best chromosome is ranked Max − (Fit + 1), and the worst chromosome is ranked 1. In this way, in this method of selecting parents, all chromosomes will have a chance to be selected.

*4.4. Crossover Operator.* The GAGELS uses a single-point exchange operator. In this operator, after selecting 2 parents for the exchange operator, a random number from the range of 1 to $n$ (which is $n$ number of operators) is selected, and the chromosome is divided into several parts equal to the selected random number. The genes in each section are then taken from the first and second parents, respectively, and copied into the offspring. Also, only one child is produced in each exchange operation. The purpose of using this exchange method is to create different children and escape from the

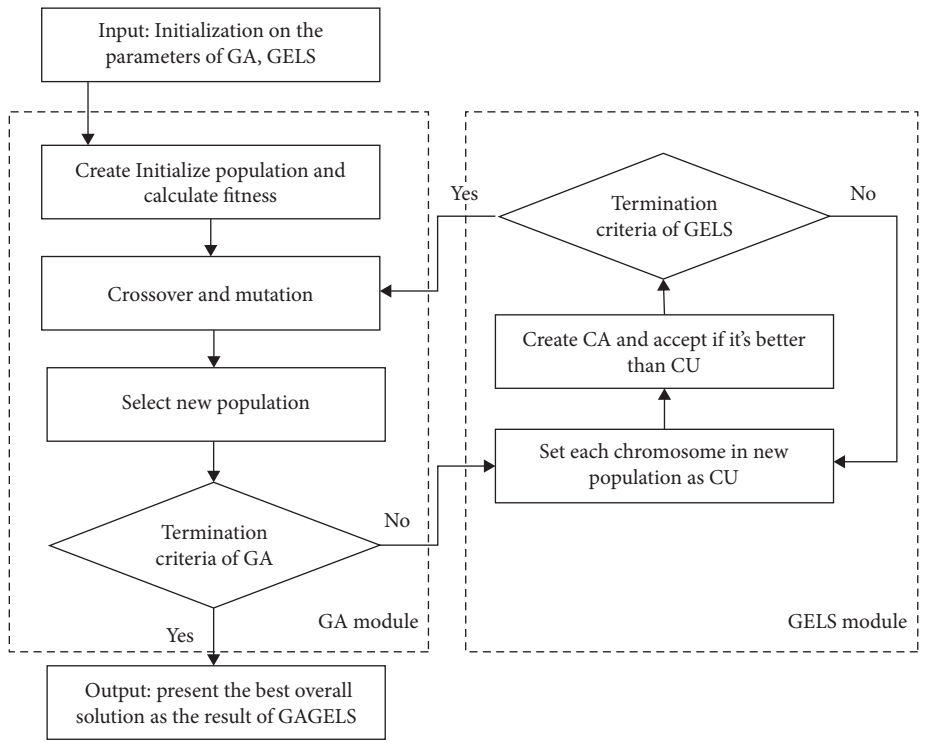


Figure 3: General steps of the proposed algorithm (GAGELS) [36].

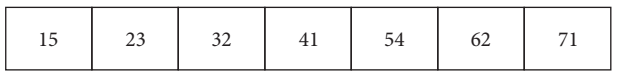


Figure 4: Illustration of a sample chromosome.

local optimality. Figure 5 shows how two different children from 2 parents are produced by the crossover operator.

*4.5. Mutation Operator.* For mutation, after selecting a parent chromosome, a gene is randomly selected, and the source of that gene is randomly changed. This type of mutation operator ensures that all possible scheduling solutions are considered according to the structure of the chromosome; in other words, the entire problem search space is covered. Figure 6 shows the actions of the mutant operator.

*4.6. Local Search with the GELS Algorithm.* Before the GELS algorithm is used to choose the chromosomes to be passed on to the next generation, a portion of the present population's chromosomes is optimized and added as new offspring. In the suggested technique, a number of chromosomes are picked, and the GELS algorithm is invoked for them in an iterative loop. In each iteration, the GELS algorithm identifies one of the population chromosomes as the current solution (CU) and optimizes it by applying its operators in many phases. In the proposed GELS algorithm to search the issue space in this manner, the CU neighborhood space is formed first, and then a mixture of the first and second approaches is used to compute the gravitational

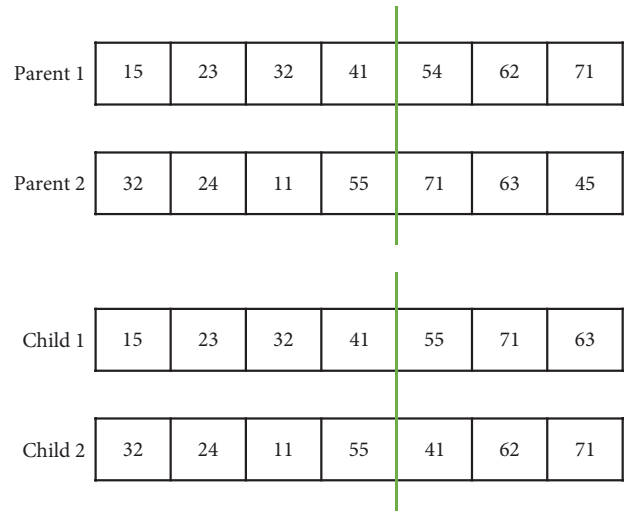


Figure 5: An example of the exchange operator for two sample chromosomes.

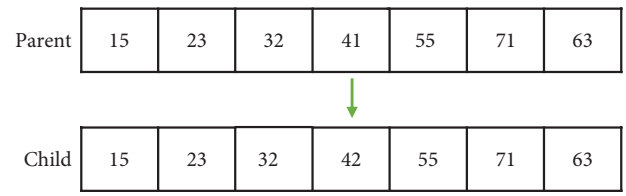


Figure 6: An example cation of the mutation operator to the sample chromosome.

Table 1: Cloud system parameters.

| Parameters | Values |
|---|---|
| Total number of tasks | 10–100 |
| Task processing time | 5–10 |
| Memory required | 50–100 |
| Number of data centers | 5–10 |
| Total memory | 250–500 |
| Number of data exchanges between different tasks | 1–10 |
| Number of data exchanges of different centers | 1–10 |

Table 2: Comparison of the results of GAGELS, PSO, and GA algorithms based on total cost.

| | Test data | GA | PSO | GAGELS |
|---|---|---|---|---|
| 1 | Data_10_5 | 491 | 1240 | 382 |
| 2 | Data_20_5 | 7829 | 8428 | 7505 |
| 3 | Data_30_10 | 11931 | 12710 | 11505 |
| 4 | Data_40_10 | 25493 | 26631 | 25164 |
| 5 | Data_50_15 | 40346 | 41117 | 40038 |
| 6 | Data_60_15 | 47713 | 49321 | 47607 |
| 7 | Data_70_20 | 56186 | 58325 | 56012 |
| 8 | Data_80_20 | 68018 | 70149 | 67923 |
| 9 | Data_90_20 | 73990 | 74511 | 73858 |
| 10 | Data_100_20 | 87113 | 88789 | 86833 |

*N*: number of tasks; *M*: number of resources.

force, with 10% of the first case and 90% of the second case being used. In addition to single and multiple, a mixture of the two is also utilized to search the search space. In the GELS algorithm, the neighborhood is defined as follows: first, the whole candidate solution (CA) is inspected from beginning to finish, and the candidate with the shortest idle time is chosen; next, by altering the sequence of its actions, an effort is made to generate a better solution. Finally, the GELS algorithm provides the best solution (BS), its optimal response to the GA. In order to optimize the chromosome as much as feasible, the optimal number of neighbors is determined. This method will be applied to every chromosome in the current population.

*4.7. Selection.* In the proposed algorithm, the chromosome selection operator for the next generation is a process of choosing from various sources. This algorithm first selects chromosomes based on merit and duplicate chromosomes, then removes 10 percent of the more suitable chromosomes, and selects the remaining chromosomes randomly for the next generation population.

*4.8. Termination Condition.* In the GAGELS, the termination condition is to limit the number of generations. That is, if the number of generations reaches the desired number, the best chromosome is displayed and the algorithm ends; otherwise, we go to step 2.

## 5. Experimental Results

We evaluated the proposed GAGELS algorithm against both the GA and the PSO technique to measure its efficacy. In our experiments, the algorithms were implemented using CloudSim software on a desktop computer with a 2.30 GHz CPU and 4 GB of RAM. Table 1 lists the cloud system parameters tested in our experiments.

For more accurate testing and comparison of algorithms, the proposed algorithm (GAGELS), GA, and PSO are tested on the same test data (*Data_N_M*). Every algorithm was run 10 times independently, and the average result was used to determine which algorithms scored the best.

Table 2 shows the total cost of the GAGELS, GA, and PSO algorithms with 500 iterations for the algorithm. According to the results, the proposed algorithm performed better than the other two algorithms.

As shown in Table 2, the numerical difference in the costs obtained from the implementation of the GAGELS of GA is less than the GAGELS and PSO because the GAGELS is based on GA and uses a GELS algorithm to optimize the answers. On the other hand, the PSO algorithm is a local search algorithm (LSA), and the probability of stopping at its local optimization is high, so it does not reach the desired answers compared to other dual algorithms.

Figure 7 shows the time required to run the algorithms. As can be seen, the proposed algorithm (GAGELS) requires more time to run than the other two algorithms. One of the reasons for the time-consuming algorithm proposed is the combination of the GA and GELS algorithms. In fact, in the GAGELS, to find better answers, we used the idea of combining the GA with an LSA based on the results of the algorithm implementation. It was suggested that this idea led to better results, but due to the combination with an LSA compared to the other two algorithms, it requires more time to run, which is the disadvantage of the GAGELS.

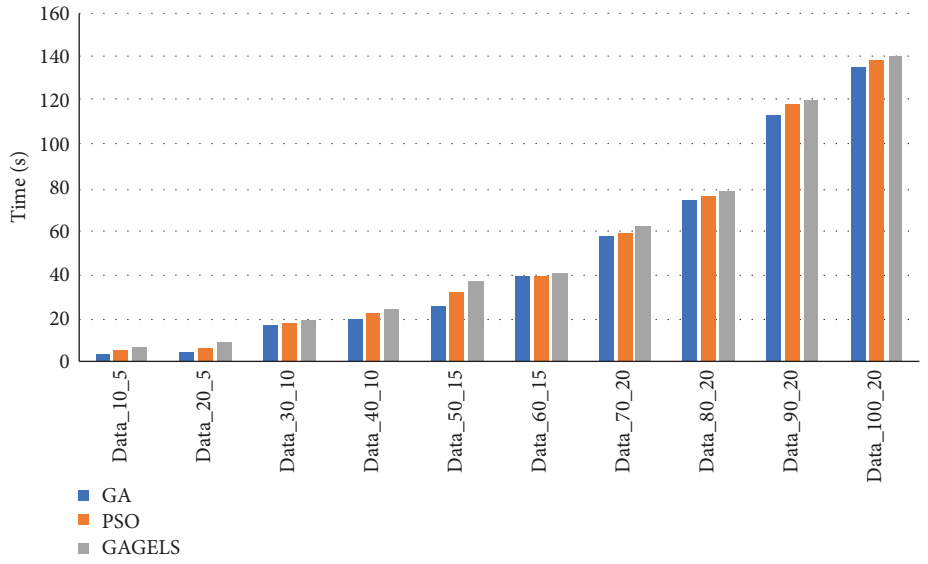


Figure 7: Time required to run GAGELS, GA, and PSO algorithms.

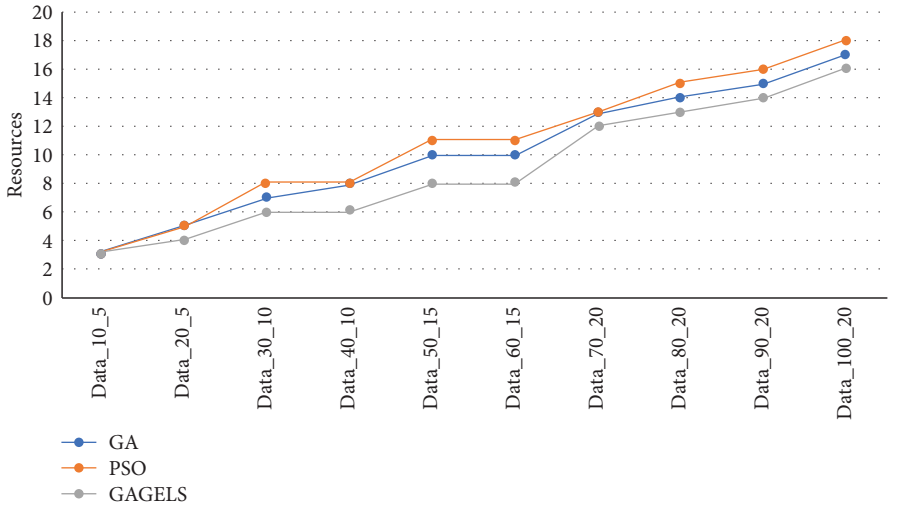


Figure 8: Number of resources allocated to the total tasks after 500 repetitions of GAGELS, GA, and PSO algorithms.

Table 3: PSO parameters [37].

| Parameters | Values |
|---|---|
| Number of iterations | 100 |
| Population size | 70 |
| Velocity bounds | [−1, 1] |
| Inertia factor $\omega$ | 0.9–0.2 |
| Acceleration constants c1 and c2 | 2 |

Figure 8 shows the maximum number of sources used by the total tasks after the completion of each load algorithm, which shows the repetition of the algorithm 200 times. Figure 8 also shows the maximum number of resources used by the total tasks after completing the execution of the algorithm with 500 repetitions. According to the figure, it can be seen that the GAGELS is more efficient than the other two algorithms in allocating resources to tasks.

Table 4: Comparison of the results of GAGELS and PSO based on total cost.

| | Test data | GA | PSO | GAGELS |
|---|---|---|---|---|
| 1 | Data_10_5 | 482 | 978 | 382 |
| 2 | Data_20_5 | 7705 | 7854 | 7505 |
| 3 | Data_30_10 | 11585 | 11692 | 11505 |
| 4 | Data_40_10 | 25197 | 25258 | 25164 |
| 5 | Data_50_15 | 40338 | 40511 | 40038 |
| 6 | Data_60_15 | 47727 | 47869 | 47607 |
| 7 | Data_70_20 | 56222 | 56370 | 56012 |
| 8 | Data_80_20 | 68129 | 68237 | 67923 |
| 9 | Data_90_20 | 74012 | 74114 | 73858 |
| 10 | Data_100_20 | 87298 | 87456 | 86833 |

In the following, we compared the GAGELS with the GA and PSO algorithms based on the parameters in Table 3. Table 4 shows the simulation results of the GAGELS with PSO and GA.

## 6. Conclusion

In this article, we provide a novel method that solves the problem of task scheduling in cloud computing by combining the GA and GELS algorithms. In fact, the proposed algorithm (GAGELS) combines the local search capability of GELS with the GA, resulting in better efficiency in reaching the best solution. The proposed algorithm's performance is evaluated by comparing its results to those of the GA and PSO algorithms. The experimental findings reveal that GAGELS is very efficient in addressing task scheduling issues and provides better outcomes. Additionally, GAGELS has 10% and 30% superiority over the GA and PSO algorithms, respectively, but it takes longer to execute. To improve the execution time of the proposed algorithm, it is feasible to assess the interaction of GA with other LSA. The influence of various GA settings on the quality of the results may be examined in future studies.

## Data Availability

No data were used to support the findings of this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References


[1] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G. B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *The Journal of Supercomputing*, vol. 76, no. 8, pp. 6302–6329, 2019.

[2] C. Liu, K. Li, K. Li, and R. Buyya, "A new service mechanism for profit optimizations of a cloud provider and its users," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 14–26, 2021.

[3] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, "Profit maximization for cloud brokers in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 190–203, 2019.

[4] K. Li, J. Mei, and K. Li, "A fund-constrained investment scheme for profit maximization in cloud computing," *IEEE Trans. Serv. Comput*, vol. 11, no. 6, pp. 893–907, 2018.

[5] J. Chen, K. Li, Z. Tang et al., "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 919–933, 2017.

[6] C. Liu, K. Li, C.-Z. Xu, and K. Li, "Strategy configurations of multiple users competition for cloud service reservation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 508–520, 2016.

[7] K. Li, C. Liu, K. Li, and A. Y. Zomaya, "Zomaya: a framework of price bidding configurations for resource usage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 8, pp. 2168–2181, 2016.

[8] R. Sakthivel, R. Kavikumar, A. Mohammadzadeh, O. M. Kwon, and B. Kaviarasan, "Fault estimation for mode-dependent IT2 fuzzy systems with quantized output signals," *IEEE Transactions on Fuzzy Systems*, vol. 298, no. 2, pp. 298–309, 2021.

[9] A. Mohammadzadeh and H. Taghavifar, "A robust fuzzy control approach for path-following control of autonomous vehicles," *Soft Computing*, vol. 24, no. 5, pp. 3223–3235, 2020.

[10] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," *Optik*, vol. 258, pp. 168677–168711, 2022.

[11] Z. Liu, A. Mohammadzadeh, H. Turabieh, M. Mafarja, S. S. Band, and A. Mosavi, "A new online learned interval type-3 fuzzy control system for solar energy management systems," *IEEE Access*, vol. 9, pp. 10498–10508, 2021.

[12] A. Mohammadzadeh, M. H. Sabzalian, and W. Zhang, "An interval type-3 fuzzy system and A new online fractional-order learning algorithm: theory and practice," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 9, pp. 1940–1950, 2020.

[13] W. Jiang, Y. Wang, Y. Jiang et al., "Mobile internet mobile agent system dynamic trust model for cloud computing," *Computers, Materials & Continua*, vol. 62, no. 1, pp. 123–136, 2020.

[14] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35–44, 2022.

[15] X. Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5603–5609, 2021.

[16] S. M. Park and Y. G. Kim, "User profile system based on sentiment analysis for mobile edge computing," *Computers, Materials & Continua*, vol. 62, no. 2, pp. 569–590, 2020.

[17] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Computing & Applications*, vol. 33, no. 19, pp. 13075–13088, 2021.

[18] M. N. Nategh, A. A. R. Hosseinabadi, S. Damavandi, and V. E. Balas, "TTGELS: a new approach for solving university exam timetabling problem by using gravitational emulation local search algorithm," *International Journal of Computational Systems Engineering*, vol. 2, no. 4, pp. 183–189, 2016.

[19] M. N. Nategh, V. E. Balas, and A. A. R. Hosseinabadi, "University-Timetabling problem and its solution using GELS algorithm: a case study," *International Journal of Advanced Intelligence Paradigms*, vol. 11, no. 3/4, p. 368, 2018.

[20] S. Selvarani and G. S. Chen, "Improved cost-based algorithm for task scheduling in cloud computing," in *Proceedings of the Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on*, pp. 1–5, Coimbatore, India, January 2010.

[21] K. Liu, H. Jin, J. Chen, X. Liu, D. Yuan, and Y. Yang, "A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform," *International Journal of High Performance Computing Applications*, vol. 24, no. 4, pp. 445–456, 2010.

[22] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *Proceedings of the Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 746-747, Washington, DC, USA, September 2011.

[23] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *The Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.

[24] H. Liu, D. Xu, and H. Miao, "Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing," in *Proceedings of the Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*, pp. 53–58, Seoul, Korea (South), February 2011.

[25] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, 2011.

[26] B. Cheng, "Hierarchical cloud service workflow scheduling optimization schema using heuristic generic algorithm," *Przeglad Elektrotechniczny*, vol. 88, pp. 92–95, 2012.

[27] M. Shojafar, M. Kardgar, A. R. Hosseinabadi, Sh. Shamshirband, and A. Abraham, "TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan", the 15th international conference on hybrid intelligent systems (HIS," *Chapter Advances in Intelligent Systems and Computing 420*, pp. 103–115, Springer, Seoul, South Korea, 2015.

[28] L. Imene, S. Sihem, K. Okba, and B. Mohamed, "A third generation genetic algorithm NSGAIII for task scheduling in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7515–7529, 2022.

[29] M. L. Chiang, H. C. Hsieh, Y. H. Cheng, W. L. Lin, and B. H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," *Expert Systems with Applications*, vol. 212, Article ID 118714, 2023.

[30] N. Manikandan, P. Divya, and S. Janani, "BWFSO: hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing," *Materialstoday Proceedings*, vol. 62, pp. 4903–4908, 2022.

[31] S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Systems with Applications*, vol. 168, Article ID 114230, 2021.

[32] A. K. Arzoo, "Hybrid ant Particle swarm genetic algorithm (APSGA) for task scheduling in cloud computing," *Information and Communication Technology for Competitive Strategies*, vol. 9-20, 2021.

[33] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Computing*, vol. 26, no. 23, pp. 13069–13079, 2022.

[34] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Multi objective task scheduling in cloud computing using Cat swarm optimization algorithm," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1821–1830, 2022.

[35] D. A. Amer, G. Attiya, I. Zeidan, and A. A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 2793–2818, 2022.

[36] A. A. Rahmani Hosseinabadi, A. Slowik, M. Sadeghilalimi, M. Farokhzad, M. Babazadeh shareh, and A. K. Sangaiah, "An ameliorative hybrid algorithm for solving the capacitated vehicle routing problem," *IEEE Access*, vol. 7, pp. 175454–175465, 2019.

[37] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 6, pp. 2370–2382, 2022.