Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 3 (2023)

IOT (Internet Of Thinks) Resource Scheduling Problem Based on Two Imperialist Competitive and Genetic Algorithms

Soheil Shakibaee

Department of Mechanics, University of Aveiro, Aveiro, Portugal,

Abstract:-The Internet of Things (IoT) is an emerging concept in the world of Information Technologies and Communications. Using the IoT in the cloud computing platform is also feasible. Appropriate use of resources such as processors is one of the challenges of the IoT in the cloud computing platform. Hence, task scheduling and IoT resources are key issues. The IoT resource scheduling is the same as the selection of an appropriate resource to equally distribute loads in processors and maximize the efficiency of resources. This study proposes a method by which the Imperialist Competitive and Genetic algorithms are combined to resolve this problem. For this, several simulations are performed on the proposed method. Simulation results indicate that the proposed method can solve this scheduling problem at an appropriate time and computational load, which could reduce energy consumption and increase the efficiency of IoT resources in a cloud computing platform.

Keywords*: IoT, resource scheduling, cloud computing, imperialist competitive algorithm, genetic algorithm*

Introduction

Cloud computing is a computational model founded on computer networks such as the Internet and provides a new pattern to provide, consume, and deliver computing services via using networks. These services include infrastructure, software, platform, and other computing resources. Cloud computing is composed of computing and cloud. By cloud, it is meant a network or a network of broad networks such as the Internet in which the user is not fully aware of what occurs there.

The working flow of cloud data centers is heterogenous due to customers of various goals and uses; thus, in a complex system of this kind, complicated scalable scheduling to cover customer needs should be expected (Zhang et al., 2015). Since the problem under consideration falls under NP-Hard problems, cloud service providers cannot provide an online decision to solve the task scheduling problem in traditional ways and at an acceptable decision time. Thus, modern heuristic methods should be utilized. There are various solutions for solving scheduling problems (Chen et al., 2012 & Zhang et al., 2015). Zhang et al. (2015) presented a scalable method for problem-solving. Although scalable, this method requires a sorting process to increase the time complexity of assigning tasks to physical machines.

In a study, Birkhoff (1946) used the heuristic BvN method to assign scheduling policies. To use such methods, it is required to understand the distribution of input processes. A scheduler should be able to maintain queue stability by maximizing the waiting time for any scheduling in a BvN decomposition matrix. A problem of this kind can be highly voluminous and increase time complexity. Also, many scheduling problems have been provided to increase gains for cloud service providers or to improve social network performance.

Conley et al. (2015) studied the properties of the working flow of public clouds. Walker (2008) compared Amazons' EC2 and several high-performance computing (HPC) cluster systems, while Mehrotra et al. (2012)

did a similar comparison between Amazon's EC2 and NASA's high-performance computing clustering system, and found that public clouds could not yield the necessary efficiency like the HPC.

In an article, Wang et al. (2010) used various benchmarks to conclude that virtual EC2 machines produced less resource efficiency while making many changes compared to specifically scientific computational clouds.

Schad et al. (2010) concluded that the heterogenous hardware of physical machines was the main factor in increasing efficiency variability. Meanwhile, Wang et al. (2014) did a study on measuring resource consumption patterns in EC2 and Azure to provide a daily consumption pattern.

Also, Koomey (2011) demonstrated that a processing time could not be a satisfactory criterion to exactly evaluate the power consumption of virtual machines and some intra-processor events affected the power consumption of processors more than others. According to some different tests, a model was proposed to evaluate the energy consumption that computed the energy consumed by a virtual machine by monitoring processors' efficiency counters.

Pietri et al. (2013) found that the Energy-Aware SLA Contract is a protocol between a cloud provider and service customers that reflects an agreed-upon domain for a cloud provider and data centers to be used in an energy-aware method and simultaneously guarantee a level of service quality for ICT customers. Today, several organizations and even people are increasingly using cloud resources.

Lordan et al. (2014) stated that scheduling resources under large-scale clouds and grids is difficult by the peer-to-peer simulations of cloud resources scheduling given the size, dynamicity, and variations of the resources. Researchers have proposed a wide range of policies and methods to map the loads presented on proposed resources. However, the current production grids are largely underperformed and do not show an effective environmental load. A simulation environment is useful for studying various methods, especially those that are expected but do not characterize the available clouds and grids. This environment is a simulator that comprises a kernel model and some mechanisms for the resources scheduling problem of a grid and describes an adaptive solution model to determine upper-performance bounds. This environment also provides a general perspective of how to use a simulator to study the performance of a grid's resource scheduling methods for various resources and load profiles under large-scale and dynamic environments.

Lucanin et al. (2015) investigated various scheduling algorithms in a cloud computing environment, concluding that cloud computing served as a provider of mobile services by using large scalable and virtual resources on the Internet.

A resource scheduling problem falls under a set of NP-Hard problems, which include several thousand various problems of various applications for which quick and feasible solutions at reasonable times have not been found. As stated, no quick solution has ever been found for them. However, it is less likely to find such algorithms. By a quick solution, it is one whose runtime is polynomially related to the input size of a problem.

The Imperialist Competitive Algorithm (ICA) is a method in the evolutionary computation domains that seeks to find an optimal answer to various optimization problems. The basic components of this algorithm consist of assimilation policies, imperialist competition, and revolution. This algorithm imitates the evolutionary social, economic, and political trends in different countries and mathematically models some parts of this process to provide operators in regular algorithmic forms, which can help solve complicated problems. In essence, this algorithm looks into answers to optimization problems in the form of countries and tries to gradually improve these answers in an iterative process and to finally get an optimized answer for the problems.

The Genetic Algorithm is a special kind of evolutionary algorithm that makes use of evolutionary biological techniques such as inheritance and mutation. The Genetic Algorithm (GA) is a programming technique that makes use of genetic evolution as a problem-solving pattern; a problem to be solved involves inputs that transform into solutions through a process model of the genetic evolution; the solutions are then assessed as candidates by an assessor function, an if the condition for the problem exists is met, the algorithm terminates.

The genetic algorithm is generally a repetition-based algorithm, most parts of which are selected in random processes.

The main goal of using cloud computing is to increase the efficiency of computing systems. To improve the efficiency of using available resources, scheduling algorithms require separating resources. Since the need for cloud IoT resources is not known at first, and also due to the variability of the needs and requirements, the utilization of those resources decreases sharply. Using meta-heuristic algorithms such as the Genetic and Imperialist Competitive algorithms causes this scheduling to occur in less time and be assigned to objects at low delays, which would result in existing machines in the IoT devices consuming less energy. This study proposed a method combining two genetic and imperialist competitive algorithms in which genetic operators are used in the repetition steps of the imperialist competitive algorithm. This combination can help perform resource scheduling with greater accuracy and speed in a cloud computing platform.

The Proposed Method

The proposed method uses a combination of genetic and imperialist competitive algorithms for a resource scheduling problem on the Internet of Things. To this aim, the proposed method uses the policy of absorption in the imperialist competition to generate new answers in the genetic algorithm. The algorithm of the proposed method is as follows:

1. Determining the chromosomal structure and scoring function (fitness)
2. Generating initial solutions and creating an initial-generation population
3. Computing the fitness of each member of a generation population
4. If the termination condition is met, the best individual is selected as the final answer and introduction to stage 8.
5. Generating new offspring using the policy of absorption of the Imperialist Competitive Algorithm (the movement of colonies to the empire)
6. Combining offspring generated by the previous population and generating a middle population
7. Selecting the best people based on the size of the population of each generation as the next generation and introduction to stage 3.
8. End

Figure 1 illustrates the flowchart of the proposed method. At first, answers are generated in the [0,1] interval using a random distribution of numbers, and then, the fitness function is used to obtain the value of each answer (individual). Later, the algorithm's termination condition must be examined to see if the condition is met or not. If yes, the proposed algorithm stops; otherwise, new offspring, which are the same new answers, should be generated. For this, the policy of absorption defined in the imperialist competitive algorithm is used. The new offspring are generated in a way to move toward the best individual in each generation, and to use the best characteristics of that individual. Then, the new offspring are combined with the previous population and give rise to the middle generation. The people in the middle generation must be sorted based on their fitness. Since the number of people in each generation must be fixed in each iteration of the algorithm, out of the people in the middle generation, selection must be made based on the number of people determined in each generation, with subsequent transfer to the next generation. This operation iterates until the termination condition is met.

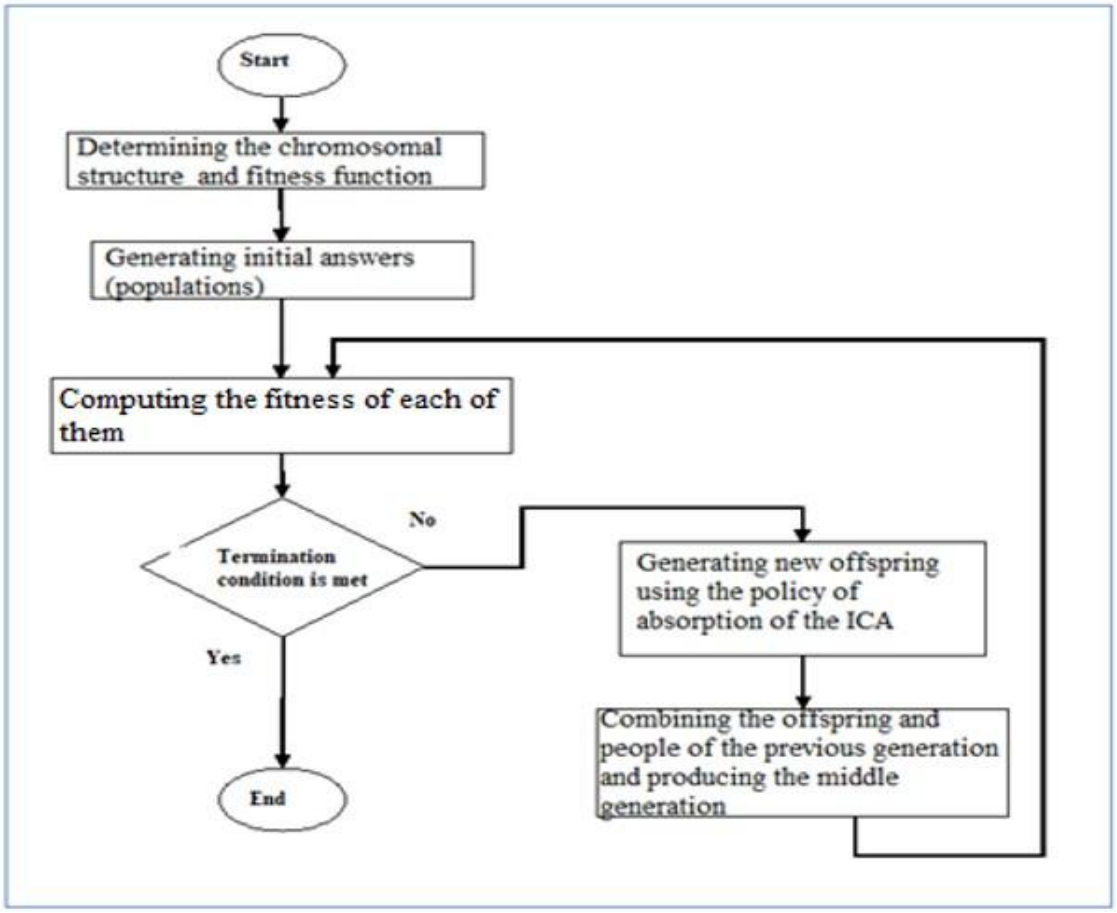

Figure 1. The flowchart of the proposed method.

In the proposed method, there are n independent tasks as resources and m processors, with each task having a certain processing time and being allowed to run on each processor for processing. All processors have the same speed. If C_{max} is assumed to be the completion time of the last task left by a system, this time must be assigned the least value possible. The fitness function, used in the proposed system, is as the following Equation (1):

$$z = \alpha \times e^{-\beta C_{max}} \quad (1)$$

Where αandβare random and optional parameters in the [0,1]interval. According to the proposed method, the scheduling problem is represented as a matrix whose rows are the number of processors in the IoT, and whose columns represent tasks assigned to processors (in line with the row number). The processing time of task *j* on processor *I* is represented by *P(i,j)*. The matrix elements are *0* or *1*, indicating the task being or not being assigned to the *ith* processor. If a *jth* task is fulfilled by an *ith* processor, the corresponding entry in the matrix will then become *x(i,j)=1*; otherwise, the entry will be *x(i,j)=0*. Equation 2 below shows the time at which tasks are completed by the processors based on the assigned scheduling.

$$C_{max} = max_{i=1}^{m}\{\sum_{j=1}^{n} x(i,j) \times P(i,j)\} \quad (2)$$

Where C_{max} represents the maximum time that takes for the entire m processor to complete a scheduled task. The goal is to minimize this value to be defined by the fitness function. Since C_{max} is a maximum function, its reverse in the objective function or its symmetry can be considered to minimize it. Thus, it is better to define it in a strictly descending (strictly monotonic) function as in the objective function in Equation (1). Each ascending or descending function is called monotonic, and each strictly ascending or strictly descending

function is called strictly monotonic. If a function is strictly monotonic, it is certainly one-to-one and thus invertible, though the opposite does not hold.

Simulations

In this simulation, the runtime of each of the input tasks on each of the processors in the IoT is given in Table 1. This simulation assumes that there are 7 tasks in the IoT, and there are 3 processors available. Each of these tasks will have different runtimes if they are assigned to each of the processors and are scheduled. For example, consistent with the first row of the following table, if Task 1 is run on Processor 1, it requires 12 units of time, and if run in Processor 2, it requires 11 units of time, while 9 units of time will be required for it to be run in Processor 3.

Table 1. Runtimes of input tasks to the IoT in Simulation 1.

Task No.	Processor 1	Processor 2	Processor 3
1	12	11	9
2	11	5	5
3	5	3	3
4	6	9	8
5	3	7	7
6	7	11	12
7	9	12	11

In this simulation, the number of iterations is 20 times, and the number of people in the population is 5.

This simulation specifies that Processor 1 runs Tasks 2 and 4, and the time to complete these two tasks is 17 units of time. Tasks 3, 5, and 6 have been scheduled on Processor 2 and the completion time equals 21 units; meanwhile, Tasks 1 and 7 have been scheduled on Processor 3, with the completion time being 20 units of time. Therefore, the completion time of all tasks based on the proposed method is 21 units. Also, the time to run the implementation code is 0.78 s. The diagram of the fitness function of the best individual in each iteration of the proposed algorithm is illustrated in Figure 2.

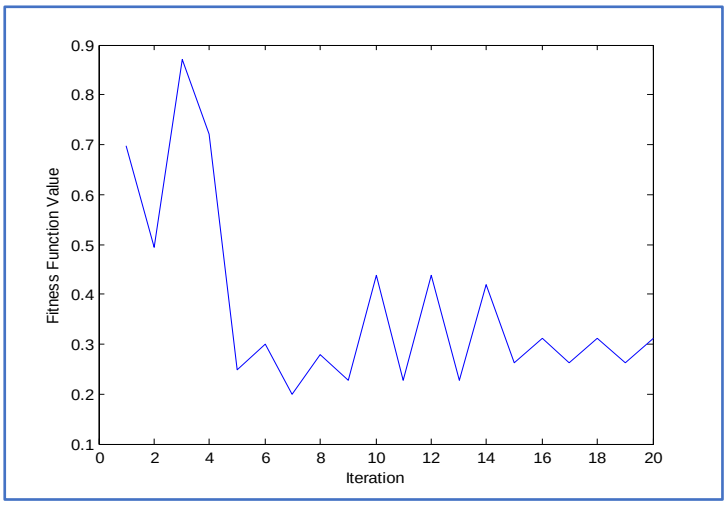

Figure 2. Fitness function in Simulation 1.

In this simulation, the number of iterations is 50 times, and the number of people in the population is 5.

This simulation specifies that Processor 1 runs Tasks 6 and 4, and the time to complete these two tasks is 13 units. Tasks 1 and 5 have been scheduled on Processor 2 and the completion time equals 18 units; meanwhile, Tasks 2, 3, and 7 have been scheduled on Processor 3, with the completion time being 19 units of time. Therefore, the completion time of all tasks based on the proposed method is 19 units. Also, the time to run the implementation code is 0.936 s. The diagram of the fitness function of the best individual in each iteration of the proposed algorithm is illustrated in Figure 3.

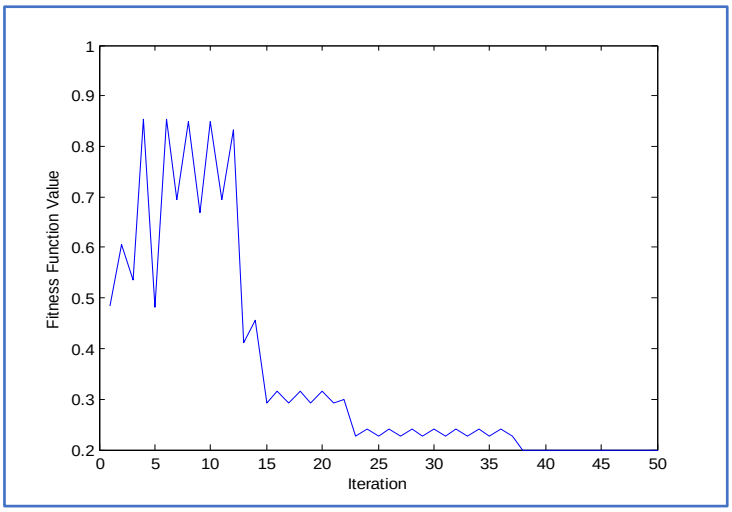

Figure 3. Fitness function in Simulation 2.

In this simulation, the number of iterations is 20 times, and the number of people in the population is 10 to examine the effects of increasing the number of people there.

This simulation specifies that Processor 1 runs Tasks 6, 4, and 3 and the time to complete these two tasks is 18 units. Tasks 1 and 5 have been scheduled on Processor 2 and the completion time equals 18 units; meanwhile, Tasks 2 and 7 have been scheduled on Processor 3, with the completion time being 16 units of time. Therefore, the completion time of all tasks based on the proposed method is 18 units. Also, the time to run the implementation code is 0.874 s. The diagram of the fitness function of the best individual in each iteration of the proposed algorithm is illustrated in Figure 4.

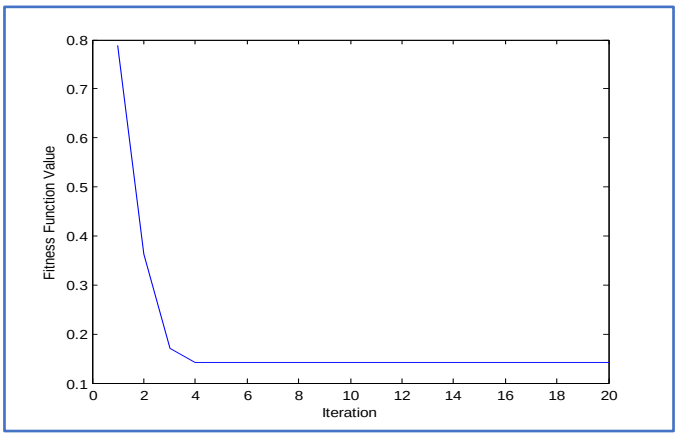

Figure 4. Fitness function in Simulation 3.

In this simulation, the number of iterations is 50 times, and the number of people in the population is 10.

This simulation specifies that Processor 1 runs Tasks 1 and 3 and the time to complete these two tasks is 21 units. Tasks 2, 3, and 5 have been scheduled on Processor 2 and the completion time equals 15 units; meanwhile, Tasks 4 and 6 have been scheduled on Processor 3, with the completion time being 20 units of time. Therefore, the completion time of all tasks based on the proposed method is 21 units. Also, the time to run the implementation code is 2.918 s. The diagram of the fitness function of the best individual in each iteration of the proposed algorithm is illustrated in Figure 5.

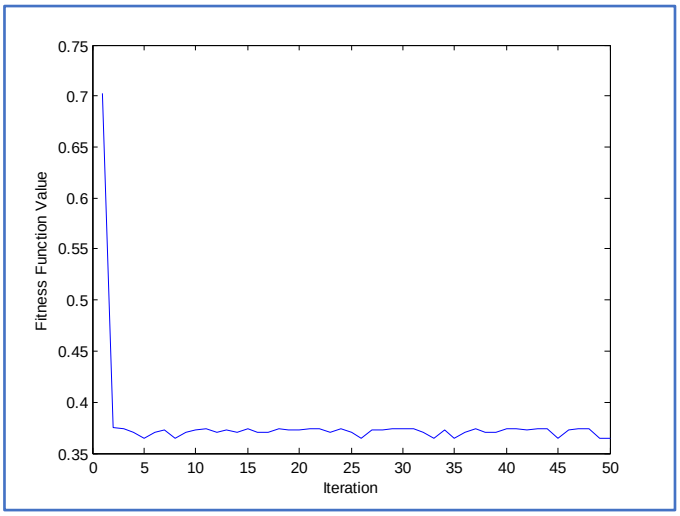

Figure 5. Fitness function in Simulations 4.

In this simulation, the number of iterations is 20 times, and the number of people in the population is 20 to examine the effects of increasing population.

This simulation specifies that Processor 1 runs Tasks 3, 4, and 6 and the time to complete these two tasks is 18 units. Tasks 2, and 7 have been scheduled on Processor 2 and the completion time equals 17 units; meanwhile, Tasks 1 and 5 have been scheduled on Processor 3, with the completion time being 16 units of time. Therefore, the completion time of all tasks based on the proposed method is 18 units. Also, the time to run the implementation code is 1.841 s. The diagram of the fitness function of the best individual in each iteration of the proposed algorithm is illustrated in Figure 6.

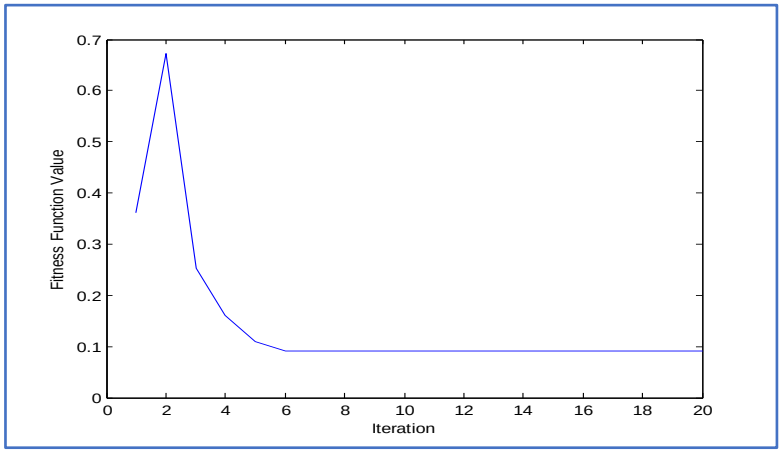

Figure 6. Fitness function in Simulation 5.

In another simulation, the proposed method is compared to the genetic algorithm method (under a standard state). For this, the number of people in the population in both methods is 10, and the algorithms are iterated 20 times. The result of this simulation is given in Table 2.

Table 2. Comparison of the proposed method with the genetic algorithm (standard state).

	Assigned tasks	**Task completion time**	**Runtime**
The proposed method	Processor 1; Tasks [3, 4 and 6] Processor 2; Tasks [1 and 5] Processor 3; Tasks [2 and 7]	18	0.874
Genetic algorithm	Processor 1; Tasks [1, 7 & 6] Processor 2; Tasks [2 and 4] Processor 3; Tasks [3 and 5]	36	0.816

In another simulation, the proposed method is compared to the genetic algorithm (under the standard state) based on different numbers of people in the population, and the completion time of the tasks for different numbers of the population is computed. For this, the number of iterated runs of the proposed method is 20 times, and the numbers of populations are 10, 20, 30, 40, and 50 individuals. The result of this simulation is given in Figure 7.

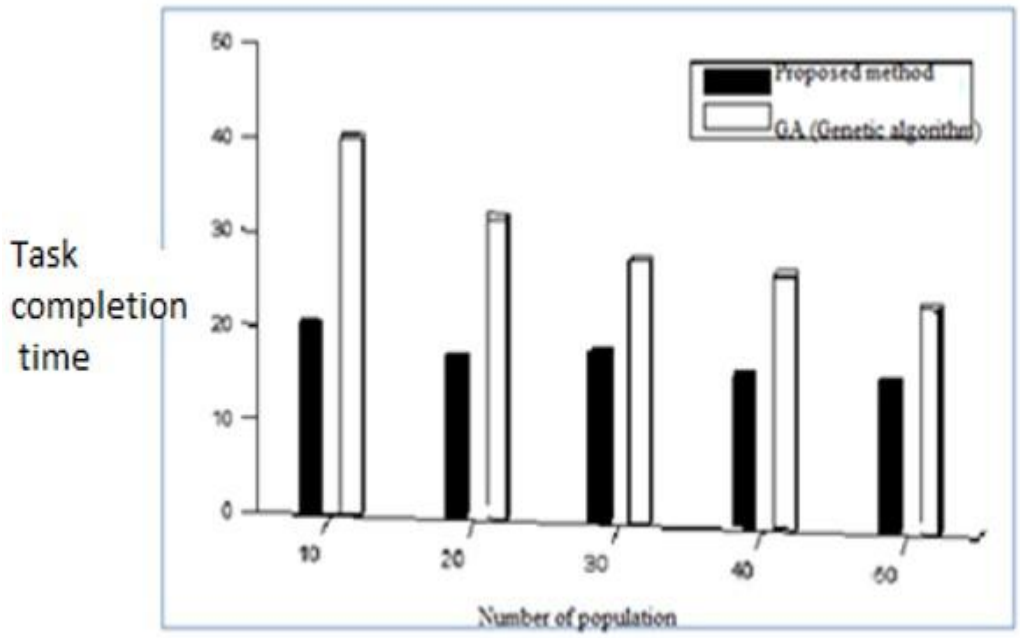

Figure 7. Task completion time for the number of different populations in the proposed method and the genetic method.

In Figure 7, the horizontal axis shows the number of different people in various runs of the proposed methods, and the vertical axis shows the time at which all tasks in the proposed and genetic methods are completed. With the rising number of populations, the time to complete the tasks decreases from 20 to 16 units in the proposed method, while it decreases from 40 to 24 units in the genetic algorithm method. Table 3 below gives a better investigation of the numbers of this diagram.

Table 3. Task completion time determined by the proposed method for the number of different populations in comparison to the genetic method.

Number of populations	Proposed method	Genetic method
10	20	40
20	17	32
30	18	28
40	16	27
50	16	24

In another simulation, the number of populations was fixed and amounted to 40 people, with the proposed method run for the number of various iterations to examine the effects of the number of iterated algorithms on the responsiveness of the proposed method. The proposed method was separately run for 10, 20, 30, 40, and 50 iterations. Also, in this simulation, the genetic algorithm-based method was run and compared to the proposed method. The result of this simulation is illustrated in Figure 8.

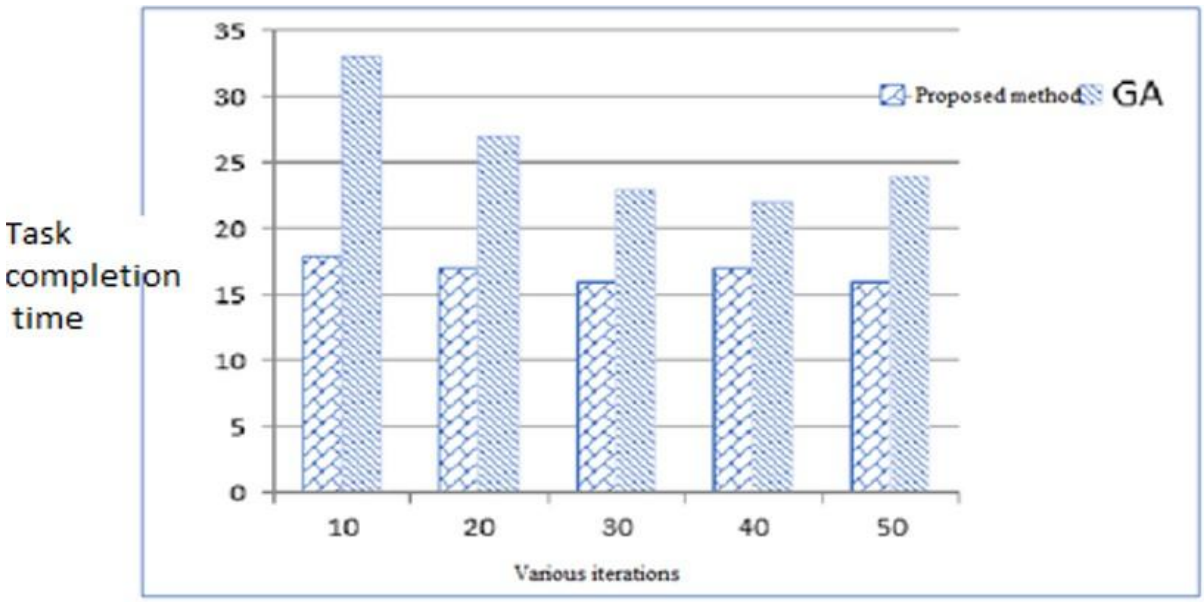

Figure 8. Task completion time determined by the proposed method for various iterations compared to the genetic method.

In Figure 8, the horizontal axis illustrates various iterations in the various runs of the proposed and genetic algorithm methods, with both methods separately running for 10, 20, 30, 40, and 50 iterations. Also, the completion time of the tasks for each run is specified by the vertical axis. The values of this diagram are given in Table 4.

Table 4. Task completion time determined by the proposed method for the number of different iterations in comparison to the genetic method.

Number of iterations	Proposed method	Genetic method
10	18	33
20	17	27

30	16	23
40	17	22
50	16	24

Conclusion

Cloud computing is used to increase computer systems' efficiency. To improve the efficiency of using available resources, scheduling algorithms require resource separation. Since the need for cloud IoT resources is not known at first, and also due to the variable of the needs and requirements, the utilization of using those resources decreases sharply. Using meta-heuristic algorithms such as the Genetic and Imperialist Competitive algorithms cause this scheduling to occur in less time and be assigned to objects at low delays, which would result in existing machines in the IoT devices consuming less energy.

This study proposed a method combining two genetic and imperialist competitive algorithms in which genetic operators are used in the repetition steps of the imperialist competitive algorithm. This combination can help perform resource scheduling with greater accuracy and speed in a cloud computing platform. The goal behind scheduling tasks and resources in the IoT in a cloud computing platform are to provide optimal scheduling for users and simultaneously provide an operational capacity for the cloud system and the QOS. Specific goals of scheduling include load balance, service quality, economic principles, the best runtime, and the system's operational capacity.

To this aim, the proposed method used the policy of absorption in the imperialist competitive algorithm to generate new answers in the genetic algorithm. Findings suggested that meta-heuristic algorithms could greatly help solve the scheduling problem of cloud resources. The proposed method can help assign tasks to processors in the IoT in a way that no processors is left without use. The proposed method helps increase the efficiency of cloud resources and reduce the answer time to users' demands.

The results of this proposed model are suggested to be examined on various types of clouds.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

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

References

[1] Birkhoff, G., 1946. Tres observaciones sobre el algebra lineal. Univ. Nac. Tucumán Rev. Ser. A 5, 147-151.

[2] Chen, F., Kodialam, M., Lakshman, T.V., 2012. Joint scheduling of processing and shuffle phases in MapReduce systems. In INFOCOM, 2012 Proceedings IEEE (pp. 1143-1151). IEEE.

[3] Conley, M., Vahdat, A., Porter, G., 2015. Achieving cost-efficient, data-intensive computing in the cloud. In Proceedings of the Sixth ACM Symposium on Cloud Computing (pp. 302-314). ACM.

[4] Koomey, J., 2011. Growth in data center electricity use 2005 to 2010. A report by Analytical Press, completed at the request of The New York Times, 9.

[5] Lordan, F., Tejedor, E., Ejarque, J., Rafanell, R., Alvarez, J., Marozzo, F., Badia, R.M., 2014. Servicess: An interoperable programming framework for the cloud. Journal of Grid Computing 12(1), 67-91.

[6] Lucanin, D., Pietri, I., Brandic, I., Sakellariou, R., 2015. A cloud controller for performance-based pricing. In Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on (pp. 155-162). IEEE.

[7] Mehrotra, P., Djomehri, J., Heistand, S., Hood, R., Jin, H., Lazanoff, A., Biswas, R., 2012. Performance evaluation of Amazon EC2 for NASA HPC applications. In Proceedings of the 3rd workshop on Scientific Cloud Computing Date (pp. 41-50). ACM.

[8] Pietri, I., Malawski, M., Juve, G., Deelman, E., Nabrzyski, J., Sakellariou, R., 2013. Energy-constrained provisioning for scientific workflow ensembles. In Cloud and Green Computing (CGC), 2013 Third International Conference on (pp. 34-41). IEEE.

[9] Schad, J., Dittrich, J., Quiané-Ruiz, J.A., 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proceedings of the VLDB Endowment 3(1-2), 460-471.

[10] Walker, E., 2008. Benchmarking amazon EC2 for high-performance scientific computing; login.The Magazine of USENIX & SAGE 33(5), 18-23.

[11] Wang, L., Nappa, A., Caballero, J., Ristenpart, T., Akella, A., 2014. Whowas: A platform for measuring web deployments on iaas clouds. In Proceedings of the 2014 Conference on Internet Measurement Conference (pp. 101-114). ACM.

[12] Wang, L., Von Laszewski, G., Dayal, J., Wang, F., 2010. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (pp. 368-377). IEEE Computer Society.

[13] Zhang, F., Cao, J., Li, K., Khan, S.U., Hwang, K., 2014. Multi-objective scheduling of many tasks in cloud platforms. Future Generation Computer Systems 37, 309-320.

[14] Zheng, Y., Shroff, N.B., Srikant, R., Sinha, P., 2015. Exploiting large system dynamics for designing simple data center schedulers. In 2015 IEEE Conference on Computer Communications (INFOCOM) (pp. 397-405). IEEE.