Pegem Journal of Education and Instruction, Vol. 15, No. 2 , 2025 (pp. 257-267)

WWW.PEGEGOG.NET RESEARCH ARTICLE

Resource Allocation in Cloud Computing Systems Using Game Theory

Somayeh Daroudi[1*]

[1] Department of Information Technology Management, Islamic Azad University, Science and Research Branch, Tehran, Iran

*Corresponding author

Abstract

Cloud computing systems consist of multiple servers that share their resources. The way these resources are utilized to execute programs affects both execution time and energy consumption. Various methods have been proposed to address this challenge, including greedy, dynamic, and metaheuristic algorithms, with the Ant Colony Optimization (ACO) algorithm being one of the most significant among them. The ACO algorithm is specifically used to find the optimal mapping of tasks to servers. Given the characteristics of metaheuristic algorithms and the inverse relationship between the number of iterations and the proximity to the optimal solution, it is crucial to prioritize these parameters to achieve better outcomes. Consequently, reaching the optimal solution may require more time. This study aims to investigate the parameters involved in the optimization and management of virtual machines in a cloud computing environment. To achieve this, simulation operations were conducted using twenty-five tasks within the CloudSim environment. Game theory was employed for the simulations, as it can yield better results than the ACO algorithm. By applying game theory, both the response time and energy consumption for the proposed tasks were reduced, leading to the development of a new method for virtual machine management.

Key words: *Game theory, cloud computing, resource allocation, player cooperation*

Introduction

Cloud computing is an emerging trend in Internet-based computing, providing resources such as energy, storage, network capabilities, and software as services. The architecture of this service consisted of three layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1]. Cloud resources are shared among multiple users and can be dynamically reallocated based on need. The clou,

How to cite this article: Somayeh Daroudi. Resource Allocation in Cloud Computing Systems Using Game Theory. Pegem Journal of Education and Instruction, Vol. 15, No. 2, 2025, 257-267

Source of support: Nil **Conflicts of**

Received: 12.11.2024

Accepted: 15.12.2024 **Published:** 01.01.2025

functions as a parallel and distributed system, consisting of interconnected virtual machines that are dynamically provisioned and presented as cohesive computing resources [2]. Virtualization is the core technology driving cloud computing. It enables great flexibility by separating physical computing resources from virtual ones, allowing them to be used independently and managed in various ways [3]. Thanks to the rapid development of virtualization technology, creating virtual machines has become a key solution for dynamic resource management within cloud computing platforms [4]. In fact, virtualization allows multiple virtual machines to operate on a single physical machine by sharing all of its hardware resources [5].

Vakiro et al. defined cloud computing as a large pool of easily usable and accessible virtualized resources, which can be dynamically adjusted and reconfigured, allowing for optimal resource utilization [6]. The National Institute of Standards and Technology (NIST) has defined cloud computing as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [6]. According to Vouk, cloud computing encompasses cyber infrastructure and is built upon concepts such as virtualization, distributed computing, grid computing, public computing, the web, and software services [7].

In one study, the Ant Colony Optimization (ACO) algorithm was used to allocate tasks to resources. This method allows ants to explore available resources but limits the number of steps each ant can take [8]. In [9], a greedy method was used to eliminate idle resources. This method cannot definitively solve the resource allocation problem due to the countless possible states that could achieve the optimal outcome. In [10], a multi-stage method was proposed that relies on predicting the load level, which necessitates accurate and reliable history- a significant challenge for the algorithm. In [11], the authors examined the scheduling problem for multi-tier web applications in virtualized heterogeneous systems to minimize energy consumption while considering the performance requirements. They concluded that the energy consumption per transaction follows a U-shaped curve, enabling the identification of an optimal efficiency point.

In [12], the authors tackled the issue of reducing energy consumption in cloud computing resources based on the defined service level agreements (SLAs) between users and service providers. They presented their study as part of a green cloud computing project, aiming to develop energy-efficient cloud resource provisioning while ensuring that quality of service requirements are met. Their focus was on the energy-aware allocation of virtual machines within cloud data centers for various application services. In [13], the authors introduced several heuristic policies for dynamically composing virtual machines. Meanwhile, in [14], the authors employed a genetic algorithm to minimize the number of calculations required by efficiently allocating resources in the cloud environment. This approach reduced overall system power consumption and achieved favorable results concerning energy efficiency and execution time. Additionally, in [8], the authors implemented an Ant Colony Optimization (ACO) algorithm to minimize response times. In this model, each ant represents a task, and resources are treated as food. Each ant moves between resources until it finds one that offers the shortest response time. The results indicated an increase in processing output and a decrease in the response time of virtual machines. In [15], the authors proposed a load balancing approach based on the Ant Colony Optimization (ACO) algorithm and complex network theory within cloud computing federations. This paper outlines a mechanism to prevent node overload and achieve load balance among nodes. When an overloaded node (meaning its workload exceeds a defined threshold) is identified, it sends out the sent "ant" (a request), and an underloaded node accepts that ant. This process enhances the load balancing between nodes and improves the overall efficiency of the mechanism. In [16], load balancing in data centers was addressed using a round-robin scheduling method. This algorithm is an energy-optimized load balancing technique that distributes tasks evenly among all virtual machines. Tasks are assigned to virtual machines in a round-robin fashion, meaning the first task is given to the first virtual machine capable of executing it. Each data center selects virtual machines locally, independently of the allocations of other virtual machines. In [17], a location-aware dynamic resource allocation model was utilized in cloud environments. This model carries out two important tasks: first, it decides where to place virtual machines, and second, it determines whether to migrate virtual machines.

In [18], a least migration policy was implemented for virtual machine migration, while a best-first algorithm was utilized for the initial deployment of virtual machines. In the best-first algorithm, having fewer virtual machines and hosts leads to better results, and the opposite is also true. In [19], the greedy algorithm was discussed. This algorithm begins by sorting all virtual machines based on a defined sorting factor, and then it calculates all possible assignments that adhere to the required conditions. The time complexity of this algorithm is O(nlogn) + O(n.m) O(nlogn) ⊂O(n.m), indicating that it is not optimal. Verna et al. [20] investigated the dynamic placement of applications within a virtualized system to minimize power consumption while maintaining SLA.

The primary challenge in distributed environments is effectively scheduling tasks and allocating resources to those tasks, which is essential for maintaining system stability [22]. Resource scheduling and allocation in cloud computing is an NP-hard problem, making it particularly important and complex, as there is no straightforward method or algorithm that can address it [23]. These processes not only impact the quality of service but also directly influence the profitability of cloud service providers. Currently, resource scheduling has become a key topic of discussion in cloud computing, and many scheduling methods have been proposed by researchers in this field. Because cloud computing resources are typically allocated dynamically and can be reclaimed, traditional methods and algorithms often struggle to accurately estimate the real requirements for resource scheduling, leading to significant resource waste.

At present, different scheduling methods are employed to assign tasks to various nodes and available virtual machines in distributed environments. While these algorithms function effectively, they still fall short in addressing certain challenges, such as communication latency, efficient resource utilization, reliability, and task assignment to unavailable machines. In the realm of cloud computing, load balancing is achieved through a process known as virtual machine migration [24].

Virtualization technology enables the online migration of virtual machines based on the load distribution across physical machines. This online migration technique allows a virtual machine to be created on one physical server and moved to another with no disruption. Some virtualization software facilitates the migration of virtual machines between different physical servers. Consequently, the primary challenge is effectively and dynamically managing the virtual infrastructure [4].

One of the most critical distinctions between **game theory** and **classical optimization theory** lies in the number and nature of decision-makers involved. In optimization models, there typically exists a **single decision-making agent** whose objective is to optimize a specific outcome or cost function. In contrast, **game-theoretic models** involve **multiple rational agents** (players), where the decision of each player **influences** the outcomes or payoffs of the others.

Consider, for instance, a consumer visiting a retail store to purchase essential goods. From a narrow viewpoint, this scenario may be modeled as a **standard optimization problem**, where the consumer aims to acquire the highest quality products at the lowest possible cost. However, a broader and more realistic perspective reveals that the **seller** is also a strategic decision-maker. The seller, in response to market dynamics, competitors' pricing strategies, supply-demand fluctuations, and customer preferences, can adjust prices or modify the product offerings.

In this context, **game theory** serves as a powerful framework to **analyze and predict the outcomes** of such multi-agent interactions. A central predictive concept within this framework is the **Nash Equilibrium**, a solution point at which no player can unilaterally improve their payoff by deviating from their chosen strategy, given the strategies of the other players. This equilibrium captures the idea of mutual best responses and provides a stable outcome in strategic environments.

This research focuses on addressing resource management issues. A key aspect of this area is the optimal use of resources and the proper allocation of tasks to the appropriate resources. Incorrect resource allocation can lead to overhead and increased energy consumption, resulting in higher carbon dioxide emissions and slower response times for cloud customers- significant problems in this field. Additionally, the efficient use of the broker, which serves as an intermediary layer, presents another challenge. The broker's role is to allocate tasks to resources, and effective management of the broker can help optimize both energy usage and response time in data centers. Response time is crucial, alongside energy consumption reduction, as it serves as an important performance parameter in this domain. The articles reviewed in this section focus on cloud computing and two types of resource allocation. A key topic of interest in these articles is virtualization, which optimally utilizes physical resources. Through virtualization, a physical resource, such as a server with hardware components like peripheral disks, main memory, and processors, can be transformed into multiple virtual servers. This research will apply game theory as a method for resource discovery in cloud computing, utilizing player cooperation to address the problem. In this framework, all players work together to maximize the overall profitability of the system. Here, tasks represent the players, and their objective is to identify the best resource for executing each task. The goal is to ensure that all tasks are completed in the shortest amount of time while minimizing energy consumption within the data center. In this system, at any given moment, players adopt one of three actions—"adding," "moving," or "displacing"- based on the chosen strategy to enhance the overall system's profitability. Accordingly, this research aims to optimize two primary parameters in real-time: energy consumption and response time in cloud computing systems.

Proposed Method

In this research, a game theory algorithm was employed to identify real-time resources for tasks aimed at reducing energy consumption and response time. The goal of this real-time resource allocation is to enhance productivity in real-world environments. For the implementation of the proposed algorithm, NetBeans software and parts of the CloudSim library were utilized.

This section discusses the fundamental concepts of rational play, focusing on dominant strategies and the elimination of dominated strategies. In this study, dominant strategies represent the most optimal allocation of machines to tasks, while dominated strategies indicate allocations that result in higher energy consumption and longer response times. In certain games, it is possible to anticipate the decisions of rational players and identify a state of the game where all players are satisfied with the current outcome. If such a state exists for the proposed game, it may be achievable through a step-by-step approach based on dominant strategies. Using this method, profiles not selected by rational players are eliminated step by step. The mathematical definition of a game can be expressed as follows:

Where strategy Si for player i is considered a dominated strategy if there exists another strategy s_i' ϵ Si for him, such that Eq. (1) holds:

$u_i(s_i'.s_{-i}) > u_i(s_i.s_{-i}); \forall s_{-i}$ (1)

where s-i represents the set of strategies adopted by the rivals of player i. This implies that optimal allocations are preferred over those that are suboptimal. By deleting the dominated strategies of players step by step, one can achieve the concept of iterative deletion of dominated strategies. If we denote the set of profiles, that were not deleted in the first step because of being undominated as D(S), the next step of deleting dominated strategies will then be confined to the set D(S). consequently, a new set denoted as D(D(D)) = D2(S) will be formed. This set is referred to as the serially undominated strategy profiles, representing the profiles that have survived the process of deleting dominated strategies. Such results can serve as initial predictions of the outcome of a game. Obviously, if players make rational decisions, there will be no profiles that include dominated strategies. However, there are scenarios where the set $D^{\infty}(s)$ contains all possible profiles. In such instances, no definitive predictions can be made regarding the outcome of the game.

Every game, much like any other optimization problem, involves a workspace that is determined depending on the problem model.

Player: In this research, the term player is attributed to tasks, meaning that the tasks defined in the problem represent the players whose goal is to find the virtual machines that provide the best efficiency for themselves and other players. Each player has characteristics such as length, input size, and output size.

Actions: As mentioned, the players aim to identify virtual machines for their operations. To this end, three actions are defined for the players:

- **Adding:** By executing this action, each player adds themselves to the tasks assigned to one of the virtual machines.
- **Moving:** By executing this action, the player changes his virtual machine. Essentially, they remove themselves from their current virtual machine and transfer to another one.
- **Displacing:** By executing this action, the player exchanges their virtual machine with another player's virtual machine, effectively transporting the other player into their current virtual machine and vice versa.

Considering the above, it can be concluded that the dimensions of the game are directly related to the number of machines and tasks and are calculated based on the number of tasks to the power of the number of machines, with the Nash equilibrium point being the primary goal of optimization through game theory. Each chosen strategy represents a combination of assigning machines to tasks, framed as a game. Each player refines their strategy based on the results they receive to guide the system toward the Nash equilibrium point, ultimately discovering more optimal solutions to the presented problem.

$GameDimension = {n_{cloudlets}}^{n_{VMs}}$ (2)

Dominated strategy

Dominated strategies refer to strategies that do not yield the optimal solution. In certain problems, this group of strategies can be identified and eliminated beforehand to diminish the problem space. For instance, strategies involving the addition of multiple tasks can be discarded to streamline the game dimensions and enhance the speed of attainment of outcomes. The dimensions of the game can thus be calculated after the elimination of the aforementioned dominated strategies:

$GameDimension = \frac{n_{cloudlets}!}{(n_{cloudlets} - n_{VMs})!}$ (3)

As seen, the dimensions of the game transition from an exponential to a factorial representation.

Dominant Strategy

Dominant strategies encompass a set of strategies that can facilitate a closer approach to the solution. These strategies contribute to a reduction in response time and energy consumption, resulting in improved payoffs.

Payoff

To compute payoffs, we take into consideration the combination of response time and energy consumption within the system.

$payoff = (w1 * ResponseTime + w2 * Energy) * vmCount$ (4)

Nash Equilibrium point

This study establishes a Nash equilibrium point by implementing a dominant strategy while avoiding any dominated strategies to maximize payoffs. At each step, the actions of "adding," "moving," and "displacing," aligned with the dominant strategy, are evaluated. The action that provides the highest operational benefit to the entire system is selected and executed. This process continues until the Nash equilibrium point is achieved.

The system under consideration is represented as a data center comprising N heterogeneous physical nodes. Each node i possesses specific characteristics: processing power (measured in MIPS), memory capacity, and network bandwidth. Tasks are assigned to virtual machines, each characterized by its processing power, memory, and network bandwidth.

The energy consumed by computing nodes in data centers is typically determined by factors such as the processor, memory, storage disks, power supply, and cooling systems.

The total power consumption of a processor is modeled using Eq. (5):

$$P_i = \beta_i + \alpha_i f(t)^3 \quad (5)$$

Where βi and αi represent idle power and proportional coefficient, respectively. In addition, f(t) denotes the frequency level of the processor at time t. Each processor is equipped with dynamic voltage and frequency scaling (DVFS), allowing it to adjust its frequency level in the range between maximum and minimum frequencies at one of the operating points. The energy consumed by the processor is calculated by Eq. (6):

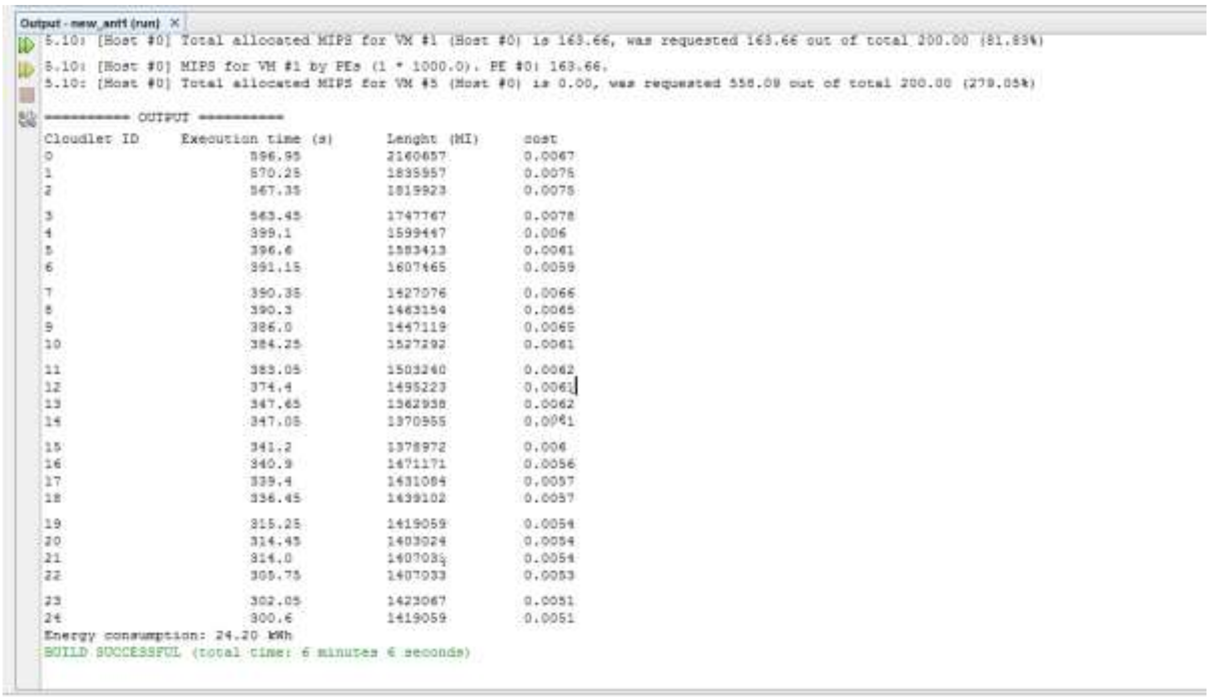
$$E_i = \int_{t_0}^{t_1} P_i\big(f(t)\big)dt \text{ , (6)}$$

The energy consumption of the processor may vary across different time intervals, as the frequency level can change between operating points.

In this research, a crucial tool called CloudSim, a Java library used for cloud development, was utilized. NetBeans software facilitates the use of this library.

Findings

In this study, ten machines were employed to test the system. Initially, results were obtained using the ACO algorithm [8], followed by results from applying game theory to enhance the system. For this purpose, a customized CloudSim environment was developed within the NetBeans platform. Figure 1 illustrates an example of the results generated by the software.



```
5.10: [Host #0] Total allocated MIPS for VM #1 (Host #0) is 163.66, was requested 163.66 out of total 200.00 (81.83%)
5.10: [Host #0] MIPS for VM #1 by PEs (1 * 1000.0). PE #0: 163.66.
5.10: [Host #0] Total allocated MIPS for VM #5 (Host #0) is 0.00, was requested 558.09 out of total 200.00 (279.05%)

========== OUTPUT ==========
Cloudlet ID     Execution time (s)      Lenght (MI)     cost
0                       596.95          2160657         0.0067
1                       570.25          1835957         0.0076
2                       567.35          1819923         0.0078

3                       563.45          1747767         0.0078
4                       399.1           1599447         0.006
5                       396.6           1583413         0.0061
6                       391.15          1607465         0.0059

7                       390.35          1427076         0.0066
8                       390.3           1463154         0.0065
9                       386.0           1447119         0.0065
10                      384.25          1527292         0.0061

11                      383.05          1503240         0.0062
12                      374.4           1495223         0.0061
13                      347.65          1362938         0.0062
14                      347.05          1370955         0.0061

15                      341.2           1378972         0.006
16                      340.9           1471171         0.0056
17                      339.4           1431084         0.0057
18                      336.45          1439102         0.0057

19                      315.25          1419059         0.0054
20                      314.45          1403024         0.0054
21                      314.0           1407033         0.0054
22                      305.75          1407033         0.0053

23                      302.05          1423067         0.0051
24                      300.6           1419059         0.0051
Energy consumption: 24.20 kWh
BUILD SUCCESSFUL (total time: 6 minutes 6 seconds)
```

Fig.1: Output of the simulator

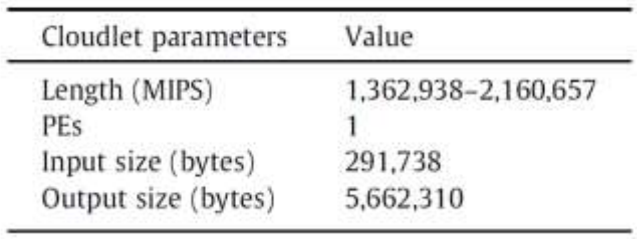
To evaluate the system, the two algorithms were tested on tasks of varying sizes, as outlined in Table 1, which details task lengths according to [8]. The unit of measurement used in this table is millions of instructions per second.

Table 1: Task length

Task No.	Task length (mips)
1	2160657
2	1835957

3	1819923
4	1747767
5	1599447
6	1583413
7	1607465
8	1427076
9	1463154
10	1447119
11	1527292
12	1503240
13	1495223
14	1362938
15	1370955
16	1378972
17	1471171
18	1431084
19	1439102
20	1419059
21	1403024
22	1407033
23	1407033
24	1423067
25	1419059

In addition to the values shown in Table 1, other parameters for the tasks must be specified, as depicted in Figure 2. These parameters, along with the task length, must be input into the simulator for execution, as indicated in [8].



Cloudlet parameters	Value
Length (MIPS)	1,362,938–2,160,657
PEs	1
Input size (bytes)	291,738
Output size (bytes)	5,662,310

Fig.2: Other task parameters

After specifying the task parameter values, the results of the simulations were analyzed. Two evaluation metrics were employed to assess the system:

Response time: This is the total time taken to send data to the virtual machine, the time required for processing, and the time taken for the data to be received by the user.

Fitness function: This function represents a value based on energy consumption and response time.

5-4) $fitness = \left[w1 \times \sum_{k=1}^{k=j} responce\ time\ + w2 \times \text{enerjy}\right] \times resourse\_count$

Response time: The time it takes for the machines to respond to the user.

Energy: The energy calculated for the tasks assigned to the machines

Resource count: Number of tasks assigned to the machines.

W1: Adjusting coefficient for response time

W2: Adjusting coefficient for energy consumption

J: The number of tasks to be performed in virtual machines

For comparison, results were extracted using input parameters for both algorithms. In the game theory algorithm, the number of players was initially set to five. Each player continued the game based on their selected strategies, with successful strategies improving the performance of all players. This strategic improvement is reflected in the fitness function shown in Table 2. At each stage, the algorithm generates new strategies based on the players' past strategies, seeking to achieve the best overall strategy for task allocation to the machines. This strategy must ultimately be the optimal one concerning both response time and fitness function, ensuring minimal response time and the lowest fitness function value across the entire game.

Table 2: Response time values for Game Theory and ACO algorithms

	Response time (s)	
Task No.	Proposed method (Game Theory)	ACO [8]
1	381	539
2	344	458
3	346	454
4	337	436
5	337	399
6	337	395
7	322	401
8	228	356
9	234	365
10	231	361
11	334	381
12	331	375
13	326	379
14	155	340
15	156	342
16	157	344
17	235	367
18	152	357
19	230	359
20	151	354
21	149	350
22	160	351
23	160	351
24	151	355
25	151	354

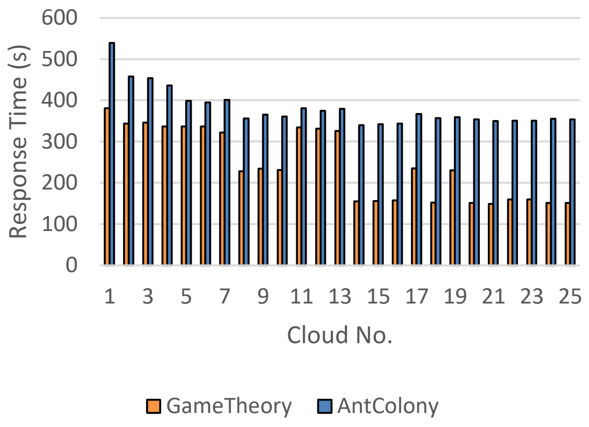


Figure 3: Response time values for Game Theory and ACO algorithms

According to Table 2 and Figure 3, it is evident that the algorithm introduced in this research has significantly decreased the response time. In most instances, it has shown results that are shorter than those of the ACO algorithm referenced in [8]. The findings related to the fitness function are presented below. The results in Table 2 are displayed in the sum of two units, seconds and watts per second.

Table 3: Fitness function values for Game Theory and ACO algorithms

	Fitness function	
Task No.	Proposed method (Game Theory)	ACO [8]
1	0.0055	0.0060
2	0.005	0.0060
3	0.005	0.0060
4	0.0049	0.0060
5	0.0049	0.0060
6	0.0049	0.0060
7	0.0047	0.0060
8	0.0033	0.0060
9	0.0034	0.0060
10	0.0034	0.0060
11	0.0049	0.0060
12	0.0048	0.0060

13	0.0047	0.0061
14	0.0023	0.0060
15	0.0023	0.0060
16	0.0023	0.0060
17	0.0034	0.0060
18	0.0022	0.0060
19	0.0033	0.0060
20	0.0022	0.0060
21	0.0022	0.0060
22	0.0023	0.0060
23	0.0023	0.0060
24	0.0022	0.0060
25	0.0022	0.0060

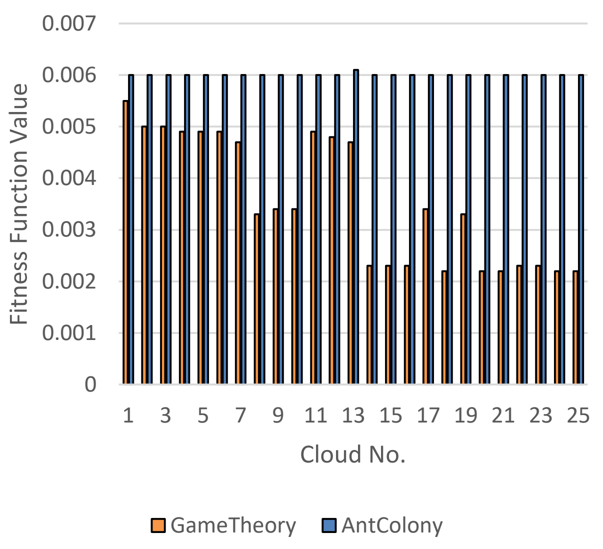


Figure 4: Fitness function values for Game Theory and ACO algorithms

Table 3 and Figure 4 display the fitness function values for the two algorithms. In most instances, the algorithm proposed in this study demonstrates a lower value compared to the ACP algorithm, highlighting its superiority.

Conclusion

This study has examined virtual machine optimization and management parameters in a cloud computing environment. For this purpose, simulations were conducted using twenty-five tasks within the CloudSim environment. In this research, game theory was utilized to achieve better results than the ACO algorithm proposed in [8]. Game theory resulted in shorter response times and less energy consumption for tasks compared to the ACO algorithm, offering a new method for managing virtual machines. By combining game theory with other algorithms, such as fuzzy logic, strategies can be developed to achieve even better results. Additionally, other meta-heuristic algorithms, such as the black hole algorithm, can be employed to expand the search space for finding solutions. This algorithm is notable for its large search space.

In order to maximize machine utilization and reduce idle times in resource allocation processes, the incorporation of additional parameters appears increasingly valuable. Among these, machine migration stands out as a promising strategy that merits deeper investigation and consideration in future research.

References

1. Gopal Kirshna Shyam, Sunilkumar S. Manvi, "Virtual Resource Prediction in Cloud Environment: A Bayesian Approach" Journal of Network and Computer Applications. March 2016
2. Saraswathi AT a, Kalaashri.Y.RA b, Dr.S.Padmavathi, "Dynamic Resource Allocation Scheme in Cloud Computing" Procedia Computer Science 47 (2015) 30 – 36
3. Lorido-Botran, T., Miguel-Alonso, J., &

Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559-592.

4. Wei-Tao Wen, Chang-Dong Wang, De-Shen Wu and Ying-Yan Xie, "An ACO-Based Scheduling Strategy on Load Balancing in Cloud Computing Environment" Ninth International Conference on Frontier of Computer Science and Technology. FCST 2015.
5. Asmae Benali, Bouchra El Asri and Houda Kriouile, "A pareto-based Artificial Bee Colony and Product Line for Optimizing Scheduling of VM on Cloud Computing" IEEE 2015.
6. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.
7. Rimal, B. P., Choi, E., & Lumb, I. (2009, August). A taxonomy and survey of cloud computing systems. In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on (pp. 44-51). IEEE.
8. Elina Pacini, Cristian Mateos, Carlos Garcia Garino, "Balancing throughput and response time in online scientific Clouds viaAnt Colony Optimization" Elsevier Science Ltd Volume 84, June 2015, Pages 31-47.
9. L. R. Moore, K. Bean and T. Ellahi, "A Coordinated Reactive and Predictive Approach to Cloud Elasticity," in Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, Spain, 2013.
10. P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in Proceedings of the 9th ACM Symposium on Operating Systems Principles, vol. 37 of ACMSIGOPS Operating Systems Review, pp. 164–177, 2003.
11. S. Srikantaiah, A. Kansal, and F. Zhao, "Energy Aware Consolidation for Cloud Computing," in USENIX HotPower'08: Workshop on Power Aware Computing and Systems at OSDI, 2008, San Diego, USA.
12. Buyya, R., Broberg, J., & Goscinski, A. M. (Eds.). (2010). Cloud computing: principles and paradigms (Vol. 87). John Wiley & Sons.
13. Beloglazov, Anton, Abawajy, Jemal, Buyy, Rajkumar," Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Elsevier Future Generation Computer Systems 28, 2012, Page(s) 755-768.
14. G. Portaluri And S. Giordano And D. Kliazovich And B. Dorronsoro, "APower Efficient Genetic Algorithm ForResource Allocation In Cloud Computing DataCenters," 2014.
15. Zhang, Z. and Zhang, X.,(2010), "A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation", Proceedings of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), pages

240- 243.

16. R. K. Mishra‹ S. Kumar‹ and B. Sreenu Naik‹ “Priority based Round-Robin service broker algorithm for Cloud-Analyst‹” Advance Computing Conference (IACC)‹ 2014 IEEE International. pp. 878–881‹ 2014.
17. Jung ,Gihun, MongSim, Kwang , "Location-Aware Dynamic Resource Allocation Model for Cloud Computing Environment" , proceedings of International Conference on Information and Computer Applications ,Dubai ,2013, Page(s)256-269.
18. T, Ferreto, M, Netto, R,Calheiros, " Server consolidation with migration control for virtualized data centers", Elsevier, Future Generation Computer systems, October 2013, volume 27 Issue 8, Page(s)1027-1034.
19. Chung Tam, Sik , Kuan, Tam , Hou, Mou Tam, Lap, Zhang, Tong, A New Optimization Method, the Algorithm of Changes, for Bin Packing Problem, IEEE, 2010, 978-1-4244-6439-5/10, 2010.
20. A. Verma, P. Ahuja, and A. Neogi, “pMapper : Power and Migration Cost Aware Application Placement in Virtualized Systems,” Ifip International Federation For Information Processing, pp. 243–264, 2008.
21. D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, “Resource pool management: Reactive versus proactive or let‟s be friends,” Computer Networks, vol. 53, no. 17, pp. 2905–2922, 2009.
22. [Gang Sun, Dan Liao1, Vishal Anand, Dongcheng Zhao, Hongfang Yu, "A new technique for efficient live migration of multiple virtual machines" Future Generation Computer Systems. September 2015
23. Weiwei Kong, Yang Lei, and Jing Ma, "Virtual Machine Resource Scheduling Algorithm for Cloud Computing Based on Auction Mechanism" International Journal for Light and Electron Optics. 2016.
24. Shridhar G.Domanal and G. Ram Mohana Reddy, "Load Balancing in Cloud Environment using a Novel Hybrid Scheduling Algorithm" IEEE 2015.