Load Balancing on the Data Center Broker Based on Game Theory and Metaheuristic Algorithms

Amine Mrhari*
Youssef Hadi†
Information Modeling and Communication Systems Laboratory, Faculty of Sciences
Ibn Tofail University
Kenitra, Morocco
**amine.mrhari@gmail.com*
†*hadiyoussef@gmail.com*

The demand for computational resources increases year over year. In recent years, resource management has become among the main research problems in the cloud computing area. In this paper, we propose two solutions based on game theory and metaheuristic algorithms: particle swarm optimization and artificial bee colony optimization to approximate the optimal or near-optimal solution for load balancing in the data center broker, based on the expected response time of users. We consider the problem as a non-cooperative game among users. The simulation results show the comparison between the proposed algorithms and demonstrate the near-optimality in terms of expected response time.

Keywords: cloud computing; load balancing; non-cooperative games; game theory; evolutionary algorithms; Nash equilibrium, particle swarm optimization, artificial bee colony optimization

1. Introduction

The use of the internet keeps increasing for serving consumers and organization needs in many areas, like communication, business and entertainment. As a consequence, high-speed processing requirements grow day after day. Cloud computing technology is a new model for the provisioning of computing resources. This paradigm shifts the location of resources to the network for reducing the costs associated with the management of hardware and software resources [1]. By employing the concept "pay-as-you-go," it offers many services and technologies over the internet, such as scalability, elasticity, mobility, availability, security and reliability [2].

Load balancing in clouds is a technique that distributes the excess dynamic local workload evenly across the various resource nodes, by

https://doi.org/10.25088/ComplexSystems.29.3.711

receiving the incoming tasks from different locations and then distributing them to the data center. It improves the overall performance of the system by achieving the optimal resource utilization ratio [3], the minimum response time and the maximum throughput. It also ensures that every computing resource is distributed efficiently and fairly [4]. It further prevents the load imbalance of the system. Load balancing also helps in the continuation of the service by a fair distribution of tasks whenever there is any service failure of one or more components. An efficient load balancing algorithm is realized if all tasks are distributed evenly between available resources. Furthermore, the algorithm must be simple, smart and scalable, if any failure happens during the runtime. Load balancing algorithms are categorized as nondeterministic polynomial hard (NP-hard) problems [5]; they can be classified as centralized or decentralized algorithms, static or dynamic algorithms. In a centralized approach, one node in the system collects all information necessary to decide the strategy for load balancing. On the other side, the decentralized algorithms, all nodes participate cooperatively or independently [6] to decide the load balancing strategy. In the static schemes, the algorithm does not use the system information to solve the problem. On the contrary, the dynamic schemes change according to the system status, which gives more flexibility to the algorithm, although it has additional costs for the system [7].

Game theory is the mathematical theory of studying the strategic interactions between the rational decision makers [8] of independent and competing actors in a strategic setting. Many solution concepts are available to formulate the rational choice. These concepts influence the outcomes of the game if the players employ the corresponding notion. In such settings, each player's decision can influence the outcomes of other players, or each player must consider how each other player will act to make an optimal choice [9]. The famous concept solution example is the Nash equilibrium. A Nash equilibrium is a selection of choices for players such that no player would prefer to unilaterally deviate from this selection.

The load balancing problem has always been a subject under study. In the literature, many studies about load balancing were proposed, and numerous approaches have been designed, such as heuristic, deterministic and game theory in many types of the environments, like grid computing [10, 11], distributed computing [6, 7], cluster computing [12, 13] and cloud computing [14, 15].

A flow chart is presented in [16] for load balancing in the cloud environment, based on the honey bee foraging strategy, for improving the processing throughput and reducing the amount of waiting time for the task on the queue as well as the response time of the virtual machines (VMs). The method of Pareto dominance's weighted sum is

used in [17] for selecting the optimal VM and running the preemptive task based on the honey bee algorithm. This method improves the system to make it more fault tolerant. Particle swarm optimization (PSO) was adopted in [18] by introducing a simple mutation mechanism and a self-adapting inertia weight method, for creating a new task scheduling model in cloud computing to avoid the load imbalance problem. It does so by improving the ant colony optimization and proposing multi-objective PSO (MOPSO) and MOPSO with importance strategy (IS) (MOPSO IS) algorithms for cloud task scheduling [19], to minimize the total task time and average task time. A novel approach for dynamic load balancing is proposed in [14]. As a result, the system performs better than the random algorithm and LBVS algorithm. The utilization of resources in the grid is improved in [10] by an adaptive and dynamic load balancing algorithm. This operation is made by associating the pheromone with resources, rather than the path and the operation of increase or decrease of pheromone. The latter represents load and depends on task status of resources. The firefly algorithm (FA) improved by [20] considered the fireflies as nodes and their attractiveness value is determined by the FA. This approach takes advantage of the constant change of population diversity to the regional optimal solution. As a consequence, the algorithm is suitable and efficient enough compared to the FA.

By including game theory concepts in the load balancing problem, the problem can be modeled as a global, non-cooperative or cooperative game [21]. In the global approach, the objective of the game is to optimize the response time of the entire system over all players by finding the social optimum [22–24]. In the non-cooperative approach, each player optimizes their own response time independently of the others and they all eventually reach an equilibrium [25–27]. Finally, in the cooperative approach, all players cooperate to make decisions by playing the optimum of each one. This gives a Pareto-optimal response time of the game and guarantees the fairness of the resources allocated to all players [27, 28].

In this paper, we consider the load balancing problem as a non-cooperative game. To approximate the optimal or near-optimal solution of the game, we based our approach on metaheuristic algorithms and the concept of Nash equilibrium.

This paper is organized as follows: Section 2 presents the load balancing scheme for cloud computing. Then, there is a discussion of the methodology for solving the load balancing problem using the non-cooperative game concept with the PSO from one side, and with the artificial bee colony optimization from the other side. Section 3 is devoted to comparing the results provided by these approaches to the non-cooperative scheme. Finally, the last section contains a summary and proposes future work.

https://doi.org/10.25088/ComplexSystems.29.3.711

2. Related Works

In this paper the load balancing problem is based on a model cited in [29], which consists of n users and m VMs. The data center is assumed to offer processing services and each VM contains one processor. The users send tasks independently to the data center with the average job rate avg_i. After the jobs arrive, the data center broker gets charged with dispatching them to VMs, taking into account minimizing the expected response time of the tasks; each VM has the average processing rate ρ_i. By integrating game theory into this problem, we supposed each user i as a player of the game with the strategy profile of job S_1; they send to the data center, where the vector $S_i = (S_{i1}, S_{i2}, \ldots, S_{im})$, and S_{ij} is the fraction of player i's job allocated on VM_j.

The load balancing strategy profile of the whole data center is presented by the matrix S, where:

$$S = \begin{pmatrix} S_{11} & \cdots & S_{1m} \\ \vdots & \ddots & \vdots \\ S_{n1} & \cdots & S_{nm} \end{pmatrix}. \tag{1}$$

Figure 1 shows the proposed load balancing problem in the cloud computing system, where θ_i is the average job generation rate of user i. Each VM is modeled as an M/M/1 queueing system [23, 30].

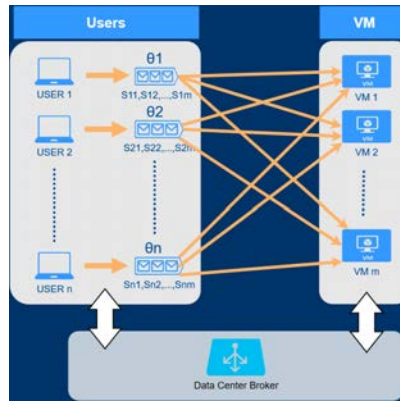

Figure 1. The proposed load balancing problem in cloud computing.

For generating a valid strategy profile for players, the following constraints must be considered [31–35]:

$$S_{ij} \geq 0 \tag{2}$$

$$\sum_{j=1}^{m} S_{ij} = 1 \tag{3}$$

$$\sum_{i=1}^{n} S_{ij}\theta_i < \rho_j \tag{4}$$

$$\sum_{i=1}^{n} \theta_i < \sum_{i=1}^{m} \rho_j. \tag{5}$$

According to [32] we introduce:

The average job rate of user i:

$$\text{avg}_i = S_i * \theta_i. \tag{6}$$

The expected response time of computer j:

$$\text{ert}_j(S) = \frac{1}{\rho_j - \sum_{i=1}^{u} S_{ij}\theta_i}. \tag{7}$$

The overall expected response time of player i:

$$\text{ort}_i(S) = \sum_{j=1}^{c} \frac{S_{ij}}{\rho_j - \sum_{k=1}^{u} S_{kj}\theta_k}. \tag{8}$$

To solve this problem, we proposed two algorithms for estimating the optimal/near-optimal strategy profile of the game, based on game theory and metaheuristics algorithms.The first algorithm, called particle swarm optimization for load balancing (PSOLB), based on particle swarm optimization, is cited in [36]. The second, called artificial bee colony for load balancing (ABCLB), based on the artificial bee colony algorithm, is cited in [37]. The results of the algorithms presented in the simulation section are compared with the non-cooperative model. In the following sections, we give a full description of the proposed algorithms.

2.1 Particle Swarm Optimization for Load-Balancing Algorithm

2.1.1 Introduction to Particle Swarm Optimization Algorithm

The PSO algorithm is a metaheuristic algorithm, based on the intelligent behavior of social organisms in groups, such as ants, birds and fish, proposed by [38]. It consists of a number of particles called a swarm. Each particle i is defined by a position vector x_i and a velocity vector v_i. At each iteration t of the algorithm, the particles are moving in the search space, based on their best personal positions and the best global position, according to the formulas:

https://doi.org/10.25088/ComplexSystems.29.3.711

$$v_i(t+1) = \omega v_i(t) + c_1 r_1\left(\text{pbest}_i(t) - x_i(t)\right) + c_2 r_2\left(\text{gbest}_i(t) - x_i(t)\right) \quad (9)$$

$$x_i(t+1) = x_i(t) + v_i(t) \cdot t. \quad (10)$$

$\text{pbest}_i(t)$ is the best personal position at the iteration t, and $\text{gbest}_i(t)$ is the global best position at the iteration t. The parameters ω, c_1, c_2, r_1 and r_2 are, respectively, the inertia weight, two positive constants and two random parameters in the range $[0, 1]$. The algorithm is iterated until the number of iterations is specified or an error criterion is reached. Figure 2 shows the movement of the particle i in the solution space during iterations t and $t+1$, and Figure 3 presents the flowchart of the basic PSO algorithm [36].

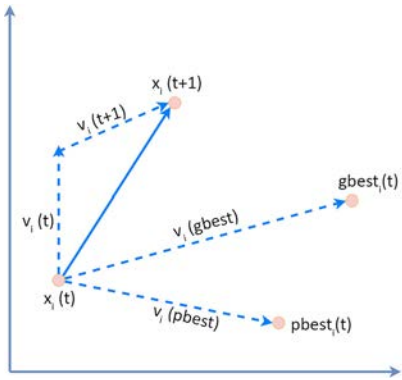

Figure 2. The movement of the particles.

2.1.2 Improved Particle Swarm Optimization to Solve the Proposed Problem

PSOLB is a PSO algorithm modified to solve the proposed game. The position of each particle on the swarm represents a strategy profile, then the algorithm is generated to find the best strategy profile for each player, by minimizing the player's expected response time, based on the Nash equilibrium concept. The output of the algorithm is the near-optimal strategy profile of the whole game.

We describe the PSOLB parameters as follows:

The velocity of particle p is the vector

$$v_p(t) = [v_1(t), (t), \ldots, v_m(t)]$$

where

$$\sum_{k=1}^{m} v_k(t) = 0.$$

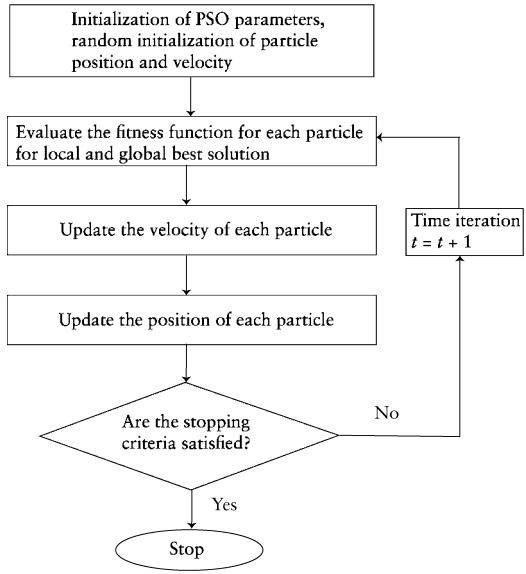

Figure 3. The flowchart of the basic PSO algorithm.

At iteration $t + 1$ we calculate as follows:

$$v_p(t+1) = \omega v_p(t) + c1 \cdot r1 \cdot \left(S_{p_{\text{per}}}(t) - S_p(t)\right) + c2 \cdot r2 * \left(S_{\text{glob}}(t) - S_p(t)\right). \tag{11}$$

Where:

- ω is a real coefficient representing the inertia weight.
- $c1$ is the personal learning coefficient.
- $c2$ is the global learning coefficient.
- $S_p(t)$ is the strategy profile of particle p at iteration t.
- $S_{p_{\text{per}}}(t)$ is the best personal strategy profile of particle p at iteration t.
- $S_{\text{glob}}(t)$ is the best global strategy profile at iteration t.
- $r1$ and $r2$ are two random numbers in the range $[0, 1]$.

The strategy profile of the particle p at iteration $(t + 1)$ is calculated as follows:

$$S_p(t+1) = v_p(t+1) + S_p(t). \tag{12}$$

Each player i on the game presents their objective function as follows, where the other players play their best strategy profiles:

$$\begin{aligned}
&\text{minimize:} \quad \mathrm{ort}_i(S)\\
&\text{subject to} \quad 0 \le S_{ik} \le 1,\ k = 1, 2, \ldots, m\\
&\qquad \sum_{k=1}^{m} S_{ik} = 1\\
&\qquad \sum_{i=1}^{n} \theta_i < \sum_{j=1}^{m} \rho_j\\
&\qquad \sum_{i=1}^{m} S_{ij}\theta_i < \rho_j\,.
\end{aligned}$$

For each player i, an NS number of particles are generated randomly, based on the function "randfixedsum" proposed by [39].

The personal best strategy for particle p:

$$S_{p_{\mathrm{per}}}(t+1) = \begin{cases} S_{p_{\mathrm{per}}}(t), & \text{if } \mathrm{ort}_p\big(SP_{\mathrm{per}}(t)\big) \le \mathrm{ort}_p\big(SP_p(t+1)\big)\\ S_p(t+1), & \text{if } \mathrm{ort}_p\big(SP_{\mathrm{per}}(t)\big) > \mathrm{ort}_p\big(SP_p(t+1)\big). \end{cases} \tag{13}$$

Where:

$$SP_{\mathrm{per}}(t) = \begin{cases} S_{p_{\mathrm{per}}}(t), & \text{if } k = i\\ S_{kj}(t), & \text{otherwise.} \end{cases} \tag{14}$$

The best global strategy profile:

$$S_{\mathrm{glob}}(t+1) = \begin{cases} S_{\mathrm{glob}}(t), & \text{if } \mathrm{ort}_p\big(SP_{\mathrm{glob}}(t)\big) \le \mathrm{ort}_p\Big(S_{p_{\mathrm{per}}}(t+1)\Big)\\ S_p(t+1), & \text{if } \mathrm{ort}_p\big(SP_{\mathrm{glob}}(t)\big) > \mathrm{ort}_p\Big(S_{p_{\mathrm{per}}}(t+1)\Big). \end{cases} \tag{15}$$

Where:

$$SP_{\mathrm{glob}}(t) = \begin{cases} S_{\mathrm{glob}}(t), & \text{if } k = i\\ S_{kj}(t), & \text{otherwise.} \end{cases} \tag{16}$$

2.1.3 The Pseudocode of the Improved Particle Swarm Optimization Algorithm

Algorithm 1. Proposed load balancing algorithm based on particle swarm optimization and game theory.

Input:
n, the number of players in the game
m, the number of VMs
ρ, the processing rate of VMs

θ, a user's job arrival rate
NP, the number of particles in the swarm
t_{max}, the maximum number of iterations
ω, the inertia weight parameters
$C1$, the cognitive acceleration parameters
$C2$, the social acceleration parameters
S, the initial strategy profile of the game
Output: The optimized strategy profile of the game.
For each player i in the game:
    Generate the initial swarm, according to the initial performances, while other players' strategies are fixed.
    Add S_i to the swarm.
    Update the personal best strategy profile and the best global strategy profile.
    While $t \leq t_{max}$ **do**
        For each particle in the swarm:
            Randomly generate $r1$ and $r2$.
            Update the velocity of the particle.
            Update the strategy profile of the particle.
            Calculate the cost function.
            Update the personal best strategy profile.
        end for
    Update the strategy profile of the swarm.
    $t = t + 1$
    end while
    Update the strategy profile of the game.
end for

2.2 Artificial Bee Colony for Load Balancing

2.2.1 The Artificial Bee Colony Algorithm

The artificial bee colony (ABC) is a metaheuristic algorithm, based on the intelligent behavior of honey bees, proposed by [37]. It consists of three essential bee groups: the employed bees, the onlooker bees and the scout bees. In the real bee colony, the employed bees search for rich food sources and share it with the onlooker bees waiting in the hive by performing a specific dance. Then the onlooker bees choose the food sources for exploiting. If the food shows a lack of nectar in relevant food sources, the food sources will be replaced by new food sources that are searched for by the scouts. In the ABC algorithm, a feasible solution to the optimization problem represents a food source, and the fitness value corresponds to the nectar of the food source. Figure 4 shows the flowchart of ABC [40].

2.2.2 Improved Artificial Bee Colony to Solve the Proposed Problem

To solve the proposed problem, we have improved the ABC algorithm proposed by [37] and we called it ABCLB, for finding the best

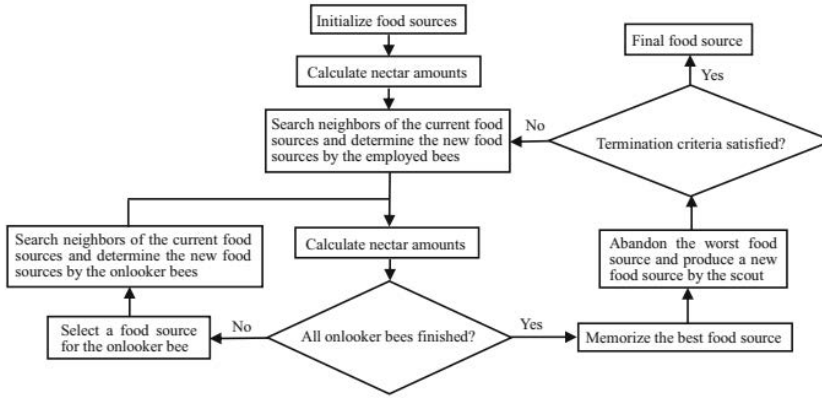

Figure 4. The flowchart of ABC.

strategy profile of the game, by minimizing the expected response time, based on the Nash equilibrium concept. The output of the algorithm is the near-optimal strategy profile. The strategy profile of the game represents the food sources, and the nectar represents the cost function. After running the algorithm for each player i in the game, an NB number of employed bees is generated. The food sources are initialized randomly according to constraints equations (2) through (5), then the nectar values are calculated according to equation (8).

We describe the ABCLB parameters as follows:

The strategy profile of the employer bee b at iteration $(t+1)$:

$$Seb_b(t+1) = S_b(t) + \left(S_b(t) - S_k(t)\right),\ k \in [1,\ b-1] \cup [b+1,\ \mathrm{NB}] \tag{17}$$

Based on the nectar values, we update the strategy profile of the bee b as follows:

$$S_b(t+1) = \begin{cases} Seb_b(t+1), & \text{if } \mathrm{ort}_i\left(Seb_b(t+1)\right) \le \mathrm{ort}_i\left(S_b(t)\right) \\ S_b(t), & \text{otherwise.} \end{cases} \tag{18}$$

The mean cost of the colony for player i:

$$\mathrm{mean}_i = \frac{\sum_{b=1}^{NB} \mathrm{ort}_i\left(S_b(t)\right)}{NB}. \tag{19}$$

The fitness values of the bee b for player i:

$$F_b = \frac{e^{-\mathrm{ort}_i(S_b(t))}}{\mathrm{mean}_i}. \tag{20}$$

The fitness values for player i are:

$$F_i = \sum_{b=1}^{NB} F_b. \tag{21}$$

Then selection probabilities of the bee b for player i are:

$$P_i = \frac{F_b}{F_i}. \tag{22}$$

Then selection probabilities for player i are:

$$P_i = \sum_{b=1}^{NB} P_b. \tag{23}$$

The best strategy profile for player i is:

$$S_{i_{\text{best}}}(t+1) = \begin{cases} S_{i_{\text{best}}}(t), & \text{if } \text{ort}_i\left(S_{i_{\text{best}}}(t)\right) \le \text{ort}_i\left(S_b(t)\right) \\ S_b(t), & \text{otherwise.} \end{cases} \tag{24}$$

2.2.3 The Pseudocode of the Improved Artificial Bee Colony Algorithm

Algorithm 2. Proposed load balancing algorithm based on artificial bee colony optimization and game theory.

Input:
n, the number of players in the game
m, the number of VMs
ρ, the processing rate of VMs
θ, a user's job arrival rate
NB, the colony size
NO, number of onlooker bees
NS, number of scout bees
L, the abandonment limit parameter
$\boldsymbol{t}_{\max}$, the maximum number of iterations
S, the initial strategy profile of the game
Output: The optimized strategy profile of the game.
For each player i in the game
    Initialize the best solution by a positive infinity number for the best cost.
    Generate the initial colony, according to the initial performances,
    while other players' strategies are fixed.
    Add S_i to the colony.
    Update the best strategy profile of the colony.
    Initialize the abandonment counter vector **C** by null numbers.
    While $t \le t_{\max}$ **do**
        for each employer bee b in the colony
            Choose k randomly, not equal to b.

https://doi.org/10.25088/ComplexSystems.29.3.711

Update and evaluate $Seb_b(t)$.
Update the strategy profile of the bee b.
end for
Calculate the mean value.
Calculate the fitness values.
Calculate the selection probabilities.
for each onlooker bee o in the colony
Select source site.
Choose k randomly, not equal to o.
Update and evaluate $Seb_o(t)$.
If it is possible, update the strategy profile of the bee I.
else $C(b) = C(b) + 1$
end if
end for
for each scout bee s in the colony
if $C(s) \geq L$
Generate new strategy profile randomly.
Update and evaluate $Seb_s(t)$.
$C(s) = 0$
end if
end for
Update the best strategy profile of the colony ever found.
$t = t + 1$
end while
Update the strategy profile of the game.
end for

3. Simulation and Results

This section gives the difference between proposed algorithms and demonstrates their efficiency by using a simulation study. First, we introduce the experimental parameters used in the simulation. Then, the experimental results are obtained. Finally, there is the evaluation of the proposed models with the non-cooperative model proposed by [7].

The simulation study was performed using MATLAB on a laptop with a Core i5 2.27 GHz processor and 4 GB RAM. We assumed the proposed system shown in Figure 1, with 3 VMs, each one containing a processor and managed by the data center broker with 10 users. Each user sends their jobs to the cloud-processing center, then the data center broker allocates the jobs to VMs immediately and sends them. Each VM receives jobs with a queueing system modeled as an M/M/1 (Poisson arrivals and exponentially distributed processing times) [22, 30], processes them and sends the results back to users. Table 1 shows the configuration information of the data center and Table 2 shows the average job generation rate of users.

VM	1	2	3
processing rate (jobs/sec)	41	47	77

Table 1. Configuration of the simulation system.

User	1-6-7	2	4-9	8-10
the average job generation rate	7	5	9	6

Table 2. The average job generation rate of users.

After running all algorithms at the same time, with the same initialization parameters as follows, we get the results shown in Figures 5 through 7.

The initialization parameters for PSOLB:

$$t_{\max} = 200, \text{NS} = 50, \omega = 0.9, c1 = c2 = 2.1.$$

The initialization parameters for ABCLB:

$$t_{\max} = 200, \text{NB} = \text{NO} = \text{NS} = 50, L = 90.$$

Figure 5 presents the values of the expected response time of users for PSOLB and ABCLB algorithms. These values are almost the same: high over 0.0327 and low over 0.0336 for PSOLB, and 0.0335 for ABCLB. The sum of expected response times of users for PSOLB and ABCLB is equal to 0.3334 and 0.3328, respectively, shown in Figure 7. That means both algorithms process jobs fairly, but ABCLB performs better than PSOLB with the value of 0.18%.

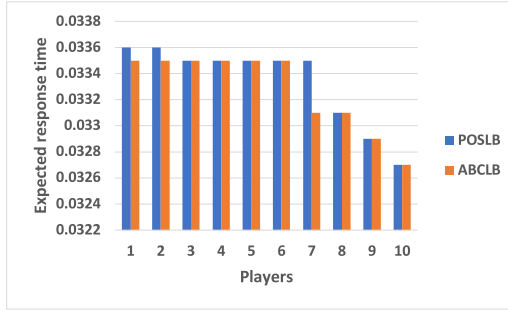

Figure 5. The expected response time of users for PSOLB and ABCLB algorithms.

Furthermore, Figure 6 compares the previous values with the values given by the non-cooperative model. These values are high over 0.0331 and low over 0.0515, and the sum of the expected response

times of users is equal to 0.4674, shown in Figure 7. That means the proposed algorithm performs better than the non-cooperative, with 28.65% and 28.78% for PSOLB and ABCLB, respectively. As a result, the algorithms estimated the near-optimal strategy of the load balancing game.

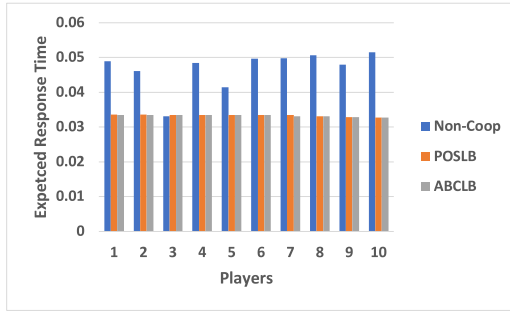

Figure 6. Comparative scheme for users' expected response times.

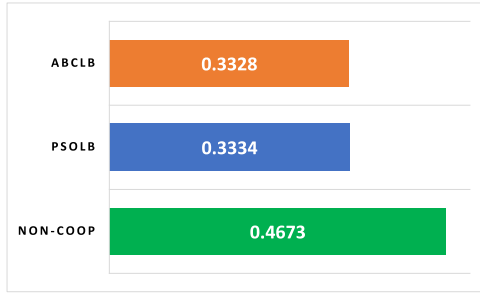

Figure 7. The sum of response times.

4. Conclusion

In this paper, we defined the concept of load balancing in cloud computing and discussed the different approaches to solving it. We proposed two algorithms to approximate the optimal solution for the response time of users by applying the concept of Nash equilibrium. These are based on the metaheuristic approach and game theory, and are called, respectively, particle swarm optimization for load balancing (PSOLB) and artificial bee colony for load balancing (ABCLB). The load balancing was modeled as a non-cooperative game among users. The simulation results showed the comparison between the

proposed algorithms and demonstrated the near-optimality in terms of expected response time for all users with fair values. Future work will aim to predict tasks in a dynamic load balancing system.

References

[1] L. Zhao, S. Sakr, A. Liu and A. Bouguettaya, *Cloud Data Management*, Cham, Switzerland: Springer International Publishing, 2014. doi:10.1007/978-3-319-04765-2.

[2] R. Moreno-Vozmediano, R. S. Montero and I. M. Llorente, "Key Challenges in Cloud Computing: Enabling the Future Internet of Services," *IEEE Internet Computing*, **17**(4), 2013 pp. 18–25. doi:10.1109/MIC.2012.69.

[3] Z. Zhang and X. Zhang, "A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation," in *Proceedings of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA 2010)*, Wuhan, China (Q. Luo, ed.), Piscataway, NJ: IEEE, 2010 pp. 240–243. doi:10.1109/ICINDMA.2010.5538385.

[4] A. M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems," *International Journal of Computer Science and Network Security*, **10**(6), 2010 pp. 153–160. paper.ijcsns.org/07_book/201006/20100619.pdf.

[5] G. W. Greenwood, "Finding Solutions to NP Problems: Philosophical Differences between Quantum and Evolutionary Search Algorithms," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 2, 2001, Seoul, South Korea, Piscataway, NJ: IEEE, 2001 pp. 815–822. doi:10.1109/CEC.2001.934274.

[6] R. Tripathi, S. Vignesh, V. Tamarapalli, A. T. Chronopoulos and H. Siar, "Non-cooperative Power and Latency Aware Load Balancing in Distributed Data Centers," *Journal of Parallel and Distributed Computing*, **107**, 2017 pp. 76–86. doi:10.1016/j.jpdc.2017.04.006.

[7] L. S. Sivalenka, Ch. Hemanandh and K. T. V. Subbarao, "The Dynamic Load Balancing Method on Game Theory for Distributed Systems," *International Journal of Science Engineering and Advanced Technology*, **2**(12), December, 2014. www.ijseat.com/index.php/ijseat/article/view/236.

[8] R. B. Myerson, *Game Theory: Analysis of Conflict*, Cambridge, MA: Harvard University Press, 1991.

[9] M. Wooldridge, "Does Game Theory Work?," *IEEE Intelligent Systems*, **27**(6), 2012 pp. 76–80. doi:10.1109/MIS.2012.108.

[10] S. K. Goyal and M. Singh, "Adaptive and Dynamic Load Balancing in Grid Using Ant Colony Optimization," *International Journal of Engineering and Technology*, **4**(4), 2012 pp. 167–174. www.enggjournals.com/ijet/docs/IJET12-04-04-026.pdf.

[11] A. V. Kale, G. U. Tangade, Y. J. Jadhao, V. P. Narkhede and S. M. Dandage, "Design and Implementation of Load Balancing in Grid Using Min-Min Algorithm," *International Journal of Research in Advent Technology*, Special Issue National Conference "CONVERGENCE 2016," 2016 pp. 73–77.

[12] P. Werstein, H. Situ and Z. Huang, "Load Balancing in a Cluster Computer," in *Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)* Taipei, Taiwan, 2006, Piscataway, NJ: IEEE, 2006 pp. 569–577 doi:10.1109/PDCAT.2006.77.

[13] S. Srivastava, P. Dadheech and M. K. Beniwal, "Load Balancing Using High Performance Computing Cluster Programming," *International Journal of Computer Science Issues*, **8**(1), January 2011 pp. 62–66. core.ac.uk/download/pdf/25821595.pdf.

[14] R. Gao and J. Wu, "Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization," *Future Internet*, **7**(4), 2015 pp. 465–483. doi:10.3390/fi7040465.

[15] A. Sharma and U. Singh, "Study on Load Balancing Techniques in Ant Colony Optimization for Cloud Computing," *IJCA Proceedings on National Conference on Next Generation Technologies for e-Business, e-Education and e-Society*, (**NGTBES 2016**)1, 2016 pp. 5–10. www.ijcaonline.org/proceedings/ngtbes2016/number1/25541-3503.

[16] R. Mishra and A. Jaiswal, "Ant Colony Optimisation: A Solution of Load Balancing in Cloud," *International Journal of Web & Semantic Technology (IJWesT)*, **3**(2), 2012 pp. 33–50. doi:10.5121/ijwest.2012.3203.

[17] C. Korat and P. Gohel, "A Novel Honey Bee Inspired Algorithm for Dynamic Load Balancing in Cloud Environment," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, **4**(8), 2015 pp. 6995–7000. doi:10.15662/ijareeie.2015.0408025.

[18] Z. Liu and X. Wang, "A PSO-Based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment," *Advances in Swarm Intelligence* (Y. Tan, Y. Shi and Z. Ji, eds.), Berlin, Heidelberg: Springer, 2012 pp. 142–147. doi:10.1007/978-3-642-30976-2_17.

[19] M. Abdullah, E. A. Al-Muta'a and M. Al-Sanabani, "Integrated MOPSO Algorithms for Task Scheduling in Cloud Computing," *Journal of Intelligent & Fuzzy Systems*, **36**(2), 2019 pp. 1823–1836. doi:10.3233/JIFS-181005.

[20] Er. M. Kaur and Er. P. Nagpal, "Cloud Computing Load Balancing Using Improved Adaptive Firefly Algorithm," *International Journal of Advances in Computer Science and Communication Engineering*, **3**(2), 2015 pp. 48–51.

[21] D. Grosu, A. T. Chronopoulous and M.-Y. Leung, "Load Balancing in Distributed Systems: An Approach Using Cooperative Games," in *Proceedings 16th International Parallel and Distributed Processing Symposium*, Ft. Lauderdale, FL, 2002, Piscataway, NJ: IEEE, 2002 p. 52. doi:10.1109/IPDPS.2002.1015536.

[22] A. N. Tantawi and D. Towsley, "Optimal Static Load Balancing in Distributed Computer Systems," *Journal of the ACM*, **32**(2), 1985 pp. 445–465. doi:10.1145/3149.3156.

[23] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed., San Diego: Academic Press, 1995.

[24] K. W. Ross and D. D. Yao, "Optimal Load Balancing and Scheduling in a Distributed Computer System," *Journal of the ACM*, **38**(3), 1991 pp. 676–690. doi:10.1145/116825.116847.

[25] H. Kameda, J. Li, C. Kim and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*, London: Springer-Verlag, 1997.

[26] C.-T. Yang, H.-Y. Cheng and K.-L. Huang, "A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud," in *GDC 2011* (T.-H. Kim, H. Adeli, H.-S. Cho, O. Gervasi, S. Yau, B.-H. Kang and J. G. Villalba, eds.) Berlin, Heidelberg: Springer, 2011 pp. 581–590. doi:10.1007/978-3-642-27180-9_70.

[27] D. Fudenberg and J. Tirole, *Game Theory*, Cambridge, MA: MIT Press, 1991.

[28] N. Sui, D. Zhang, W. Zhong, L. Wu and Z. Zhang, "Evolutionary Game Theory Based Network Selection for Constrained Heterogeneous Networks," in *2015 2nd International Conference on Information Science and Control Engineering*, Shanghai, China, Piscataway, NJ: IEEE, 2015 pp. 738–742. doi:10.1109/ICISCE.2015.170.

[29] A. Mrhari and Y. Hadi, "A Load Balancing Algorithm in Cloud Computing Based on Modified Particle Swarm Optimization and Game Theory," *2019 4th World Conference on Complex Systems (WCCS)*, Ouarzazate, Morocco, Piscataway, NJ: IEEE, 2019 pp. 1–6. doi:10.1109/ICoCS.2019.8930807.

[30] R. Jain, *The Art of Computer Systems Performance Analysis*, New York: Wiley, 1991.

[31] S. Penmatsa and A. T. Chronopoulos, "Game-Theoretic Static Load Balancing for Distributed Systems," *Journal of Parallel and Distributed Computing*, **71**(4), 2011 pp. 537–555. doi:10.1016/j.jpdc.2010.11.016.

[32] H. Siar, K. Kiani and A. T. Chronopoulos, "An Effective Game Theoretic Static Load Balancing Applied to Distributed Computing," *Cluster Computing*, **18**(4), 2015 pp. 1609–1623. doi:10.1007/s10586-015-0486-0.

[33] D. Grosu and A. T. Chronopoulos, "Noncooperative Load Balancing in Distributed Systems," *Journal of Parallel and Distributed Computing*, **65**(9), 2005 pp. 1022–1034. doi:10.1016/j.jpdc.2005.05.001.

[34] V. Mani, S. Suresh and H. J. Kim, "Real-Coded Genetic Algorithms for Optimal Static Load Balancing in Distributed Computing System with Communication Delays," in *Computational Science and Its Applications (ICCSA 2005)*, Singapore (O. Gervasi, et al., eds.), Berlin, Heidelberg: Springer, 2005 pp. 269–279.

[35] X. Tang and S. T. Chanson, "Optimizing Static Job Scheduling in a Network of Heterogeneous Computers," in *Proceedings of the 2000 International Conference on Parallel Processing*, Toronto, Canada (D. J. Lilja, ed.), Piscataway, NJ: IEEE, 2000 pp. 373–382. doi:10.1109/ICPP.2000.876153.

[36] R. Umar, F. Mohammed, M. Deriche and A. U. H. Sheikh, "Hybrid Cooperative Energy Detection Techniques in Cognitive Radio Networks," *Handbook of Research on Software-Defined and Cognitive Radio Technologies for Dynamic Spectrum Management* (N. Kaabouch and W.-C. Hu,, eds.), Hershey, PA: IGI Global, 2015 pp. 1–37. doi:10.4018/978-1-4666-6571-2.ch001.

[37] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report-tr06, Computer Engineering Department, Erciyes University, Kayseri, Turkey, 2005.

[38] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 International Conference on Neural Networks*, Perth, WA, Australia, Piscataway, NJ: IEEE, 1995 pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[39] R. Stafford. "Random Vectors with Fixed Sum." (Jun 18, 2020). www.mathworks.com/matlabcentral/fileexchange/9700.

[40] L. Zhang and N. Xiao,"A Novel Artificial Bee Colony Algorithm for Inverse Kinematics Calculation of 7-DOF Serial Manipulators," *Soft Computing*, **23**(10), 2019 pp. 3269–3277. doi:10.1007/s00500-017-2975-y.