DOI: 10.14529/jsfi230101

An Ant-colony Based Model for Load Balancing in Fog Environments

Seyedeh Leili Mirtaheri[1], *Mahya Azari*[2], *Sergio Greco*[3], *Ehsan Arianian*[4]

© The Authors 2023. This paper is published with open access at SuperFri.org

Delay-sensitive applications are becoming more and more in demand as a result of the development of information systems and the expansion of communication in cloud computing technologies. Some of these requests will be overlooked in cloud environments due to the communication delay between the processing center and the client's request. The 'fog-based computing paradigm', a novel processing model, can be added to cloud computing to help with the aforementioned issues. The performance of computing systems is always influenced by latency. In this paper, we focus on balancing the load on the fog nodes to lower the latency. It is also a crucial component of fog computing devices. It necessitates the use of load-balancing algorithms to select the optimal hosts, resulting in an even distribution of the load on the available resources. We provide a load-balancing approach based on the Ant-colony optimization algorithm's latency rate for responding to tasks. A random data set evaluation of this model reveals shorter response times than those of earlier strategies suggested in this field.

Keywords: fog environment, cloud computing, load balancing, Ant-colony.

Introduction

One of the biggest problems in the distributed computing environment is load balancing. Numerous requests made by thousands of users and customers necessitate the utilization of a lot of hardware and bandwidth. A load balancer assists in distributing the burden among the many nodes and making sure that none of them are overwhelmed. It also expedites the completion of the most recent work among the sources that use the phrase [38]. Distributed systems like fog and cloud computing need an effective load-balancer.

Delivery time also dramatically increases as the number of delay-sensitive requests and the computational load on fog nodes both rise. As a result, some nodes must assign jobs to nodes that are less busy. Fog computing systems, on the other hand, have several connected processors that work independently. Every processor has a distinct processing capacity and an initial load. To reduce processing time and CPU downtime, the load is spread among processors based on their processing speeds. Since load balancing is meant to boost performance, its absence in distributed systems is a major issue [15, 48].

Environments that are both homogenous and diverse present a problem for load balancing methods in fog computing. In a homogenous group, all fog nodes have the same capacity and characteristics. Because they are unable to make use of the heterogeneous nature of resources, these nodes rarely adapt to fog settings. Each fog node in a heterogeneous group has a unique capacity, set of processing capabilities, and set of attributes [11]. Depending on the demands of the jobs, the load balancer can choose several nodes. In our suggested paradigm, we presume that all fog nodes are heterogeneous and that each fog node has distinct capacities and hardware architecture.

[1]Corresponding Authors, Department of Electrical and Computer Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran. Mirtaheri@khu.ac.ir
[2]Faculty of Mathematics and Computer Science, Kharazmi University, Tehran, Iran
[3]Department of Computer Engineering, Modeling, Electronics and Systems, University of Calabria, Italy
[4]Department of Information Technology, ICT Research Institute (ITRC), Tehran, Iran

The nature of the tasks has a big impact on how well load-balancing algorithms work. Therefore, the greater optimization would be possible, the more information about the tasks would be available at the time of decision-making. Having a general understanding of job execution times and the precise due date for each task increases the likelihood of obtaining the ideal load distribution. Having a set time for each activity to be completed is uncommon; yet, this is an ideal condition. For this reason, a variety of strategies have been taken into consideration for organizing and generating thoughts for the duration of task execution. The maximum execution time needed for each task is taken into account in our proposed approach during load balancing.

An optimization strategy must be used since selecting the optimum hosts (load distribution on resources) is an endless sort of problem with multiple viable solutions. In this study, we employ the Ant-colony algorithm and the meta-heuristic methodology. Glover introduced the term "meta-heuristics" in 1986 in order to help people solve difficulties by identifying solutions that are close to ideal [19]. The meta-heuristic algorithms use governing mechanisms that mimic certain methods found in nature, social behavior, physical laws, etc. to achieve complete exploration of the search space. Meta-heuristic algorithms begin with an initial collection of independent variables and evolve to find the objective function's global minimum and maximum. We encountered the similar issue when balancing the system's load, thus we used one of the well-known meta-heuristic algorithms, called Ant-colony, in our proposed model.

We start with balancing the workload between tasks and fog nodes as input for the issue in our suggested model. Depending on the nature of the problem, the list of inputs may change. The current study takes into account a list of tasks with due dates and set intervals for each request. Taking into account the distance to the fog node, each assignment must be completed within the allotted amount of time. You might categorize these lists as static or dynamic. Requests are posted online to the computing system in the dynamic list. In contrast, it is believed that we have a collection of tasks whose order is flexible in the static model. The current study makes the assumption that there are a number of delay-sensitive jobs that ought to be.

It is crucial to know how many fog nodes there are and which ones are capable of carrying out the necessary activities. The list of fog nodes is supplied in our study so that the capacity, traits, and processing power of each node can be noted. The solution to load balancing issues in a fog environment depends on the problem's objective function. The goal of the issue can be to raise the quality-of-service metrics (QoS), which include optimum resource utilization and decreased throughput, costs, response times, latency, and energy usage. To assess how well the problem satisfies the established aim, the problem's goal must be modeled. The goal of the current work is to decrease latency in order to speed up task reaction times.

In the following sections of the present paper, we first explained related work. The current definition of load balance, types of load balancing mechanisms, and reasons for its need, benefits, and criteria of load balance in fog environment. The Ant-colony algorithm will be explained according to the classification of previous models. The proposed model is introduced and discussed, and lastly, the research results will be presented.

1. Related Work

Using optimization models, the load balance issue in fog situations can be assessed. Using meta-heuristic models to offer close to optimal solutions for such issues in an acceptable amount of time is one method to problem solving. Large-scale, complicated problems are thought to be amenable to the use of meta-heuristic models, which fall into two categories:

- The first category: is based on individuals and modifies a candidate solution.
- The second category: is population-based and improves several solutions.

In addition, biological behaviors can be categorized based on process strategies in the classification of reproductive strategies or inspired by natural phenomena. Given those various mechanisms have been proposed for load balancing in a fog environment. Some of these mechanisms are based on meta-heuristic algorithms, such as the bee algorithm, particle swarm, hill-climbing, and Ant-colony algorithm. Ant-colony Optimization algorithm is one of the most popular optimization algorithms with the ability to adapt to the system environment and the flexibility to solve any optimization problems. The present research uses Ant-colony algorithms to balance the load in the fog computing environment [43].

Zahid et al. [57] discussed a Hill Climbing technique to manage the load on VMs with the aim of reducing the response time, processing time, and delay. In the proposed framework, four different regions are taken, with each region containing a fog with a cluster of buildings. Results from the simulation reveal that the proposed strategy performs satisfactorily.

Kamal et al. [25] addressed the issue of load balancing by employing a heuristic approach for solving a Constraint Satisfaction Problem (CSP). The algorithm first checks the assignment conflicts and then randomly assigns VMs to requests. The proposed approach is compared in terms of processing time, response time, and cost with Throttled and Round Robin algorithms. It helps in allocating optimal resources to requests, thereby providing better simulation results.

Naqvi et al. [35] has proposed a bio-inspired algorithm called Ant-colony optimization for load balancing. The Ant-colony optimization algorithm is the swarm-based genetic algorithm. It works on the mechanism of real ants using pheromones in order to explore their path. In the same way, the allocation path of the cloudlets is identified based on the minimal path cost in a probabilistic manner.

In the paper [29], a meta-heuristic scheduler Smart Ant-colony Optimization (SACO) task offloading algorithm inspired by nature is proposed to offload the IoT-sensor applications tasks in a fog environment. The proposed algorithm results are compared with Round Robin (RR), throttled scheduler algorithm, and two bio-inspired algorithms such as modified particle swarm optimization (MPSO) and Bee life algorithm (BLA). The numerical result shows the significant improvement in latency by the proposed Smart Ant-colony Optimization (SACO) algorithm in task offloading of IoT-sensor applications comparison to Round Robin (RR), throttled, and MPSO and BLA. The proposed technique reduces the task offloading time by 12.88, 6.98, 5.91, and 3.53% in comparison to Round Robin (RR), throttled, MPSO, and BLA.

In this study [40], utilizing a hybrid optimization algorithm, they introduced a novel energy-aware technique for load balance management in the fog-based VANET. A VANET network is made up of mobile nodes without any infrastructure. The mobility of nodes, limited energy reserves, lack of central management, and providing a guarantee for the quality of services are some of the challenges of this type of network compared to wired networks. The existing nodes of VANET networks are free to move in any direction. This study aimed to present an energy-aware model based on load balancing using the ACO algorithm. Algorithm implementation and routing models were fully described. Then, the effect of increasing the number of nodes on the amount of energy consumed by the fog-based VANET and the residual energy in the battery were investigated. Also, the number of residual nodes was increased by increasing the length of routing periods in the proposed model with the ant and ABC optimization algorithm.

The simulation results in the NS2 environment showed that with increasing the number of nodes, the amount of energy consumed by the fog-based VANET increases. Also, as the number of nodes increases, the amount of residual energy will decrease, making the proposed algorithm outperform the other two algorithms. Examining the amount of residual energy within the periods revealed that the amount of residual energy decreased with increasing periods. The results indicate the good performance of the proposed model compared to the other four models.

In this paper [5], the cost-aware Ant-colony optimization-based load balancing model is proposed to minimize the execution time, response time, and cost in a dynamic environment. This model enables to balance of the load across the virtual machines in the data center and evaluates the overall performance with various load-balancing models. As an average, the proposed model reduces carbon footprint by 45% as compared to existing models.

In this paper [20], they propose a multi-objective fog computing task scheduling algorithm based on an improved Ant-colony algorithm, which optimizes the Ant-colony algorithm to make it more suitable for the characteristics of the fog node, uses time and cost (TAC) to comprehensively consider the cost of the node, and introduce the critical factor in task allocation to improve the convergence speed of the algorithm. Different simulation experiments show that the efficiency of the improved Ant-colony algorithm is enhanced in processing time, cost, and load balance.

In this research [7], they have proposed a new algorithm called Ant-colony Optimization-based Light Weight Container (ACO-LWC) load balancing scheduling algorithm for scheduling various process requests. Initially, the task allocation was done in a round-robin fashion. The CPU usage and memory usage were maintained within a particular optimal range to achieve effective load balancing in this algorithm. Here, the load balancing scheduling was done based on Ant-colony Optimization. Performance analysis showed that the proposed ACO-LWC algorithm achieved better stability performance. Further, the TPS of the cluster for 50 applications was found to be 120, 231, and 1042 for the least connection, round-robin, and proposed ACO-LWC, respectively. Similarly, the response time for the least connection, round-robin, and the proposed scheme with 60 applications was identified as 5282 ms, 2865 ms, and 1593 ms. Furthermore, analysis shows that the overall run time is very low for the proposed scheme. There is around 2.08 times reduction in run time compared to the least connection and 1.607 times reduction in run time compared to the round-robin algorithm.

2. Proposed Model

In this section, we describe the process of using the Ant-colony algorithm in load distribution problem in our proposed model.

2.1. Ant-colony Algorithm

The Ant-colony optimization model is based on the actual movement of ants in nature. Ant-colony optimization algorithms are inspired by natural phenomena and biological behaviors and approach finding near-optimal solutions using a probabilistic model [16].

2.2. Problem Inputs

The problem inputs are mentioned in the following:

- List of fog nodes: This list includes the processing power of each node in terms of the number of cycles per second (cycle/s). Along with this list, the capacity of each fog node is also considered.
- To-do list: This list includes tasks that must be processed in fog nodes. This list is included: the minimum processing power required in terms of bits per second, the volume of tasks in megabytes, the distance of the task producer service from the surrounding fog nodes in meters, and the deadline of each task.
- Fixed parameters: In the Ant-colony optimization algorithm, several fixed parameters are used, the values of which are selected based on [58].

2.3. Latency Model

Latency is the time that each processor spends to process requests, i.e., from the moment the request is transmitted until the customer receives the request after the processing. For nodes in the fog environment that are close to the end devices, it is possible to collect data from sensors / IoT devices and process, analyze, and store them on the network edge. It should be done with minimal delay. The present study aimed to minimize the latency of each task on the fog nodes [28]. The completion time for each task includes the execution time of each task and its transmission time. The execution time of task i on node j is calculated using equation (1) [45]

$$TE_{ij}(t) = \frac{task\_size_i \cdot task\_proc_i}{fnode\_proc_j}. \tag{1}$$

In this equation, $task\_size_i$ indicates the size of task i, $task\_proc_i$ indicates each task's capacity, and $fnode\_proc_j$ indicates the capacity of the node j. At any given time (t), the transfer time, which is the time spent for sending task i to node j, is calculated using equation (2)

$$TR_{ij} = \frac{task\_size_i}{task\_dist_{ij} \cdot task\_deadline_i}. \tag{2}$$

In this equation, $task\_dist_{ij}$ indicates the distance of task i to node j, and $task\_deadline_i$ indicates the specified time for each task. Lastly, the total delay is calculated using equation (3)

$$total\_delay_{ij}(t) = TR_{ij}(t) + TE_{ij}(t). \tag{3}$$

2.4. Mapping of Load Balancing Parameters on the Ant-colony Parameters

As mentioned at the beginning of this section, the load balancing problem uses delay time to reduce the response time. In the case of load balancing in the target fog environment, the load balance between the fog nodes is set so that the objective function, which is the rate of delay, is met and minimized. It can be described as assigning n independent tasks to m fog nodes. Ants place a list of tasks for processing on the fog nodes. Each task has specific processing power, size, and distance from the fog nodes, and also, based on the specific time limit set for each task, the tasks can be executed on their appropriate fog node. For each fog node, a certain amount of processing power is considered in $cycle/s$. As mentioned before, the Ant-colony algorithm is an optimization algorithm, and each optimization algorithm has a certain number of iterations in which the problem is led toward the optimal solution.

2.5. Quantification of the Initial Solution

Every meta-heuristic optimization problem requires starting with an initial solution to explore the search space. The first step in the optimization algorithm is to create an initial solution to start the optimization process. In the Ant-colony algorithm, like other meta-heuristic algorithms, the initial solution must be applied as one of the inputs to the problem. Initial quantification of pheromones is also performed at this stage. The initial pheromone is given to prevent the ants from staying in place and not moving at the starting point. As a result, the initial solution is made using Initial quantification. In the present study, the initial number of pheromones is determined based on the random assignment of each task to the node, and their delay time is calculated. The pheromone left by the ants along the way indicates the tendency of the tasks toward the fog node.

2.6. Pheromone Factor

The number of pheromones in the different pathways drives the ants to reach the goal. In the ant pheromone cloning algorithm, pheromones indicate the desirability of the path. In our proposed model, the delay time factor is used in place of pheromones. The higher the pheromone, the shorter the latency in assigning task i to node j. Equation (4) introduces the pheromone factor

$$\tau_{ij} = \frac{1}{\varepsilon \cdot delay_{ij} + (1-\varepsilon) \cdot dist_{ij}}, \tag{4}$$

where ε is a constant parameter, and its value is determined [58].

2.7. Heuristic Information

Apart from pheromone pathways, another vital factor in the Ant-colony optimization algorithm is the selection of a suitable heuristic, which will be used in combination with pheromone information to construct solutions. At each stage of constructing a solution, the candidate selected for transfer depends on two factors: the pheromone factor and the heuristic factor. Since innovative information is calculated for all the movements in all the ants, it is, therefore, an essential factor that affects the performance of the Ant-colony algorithm. The heuristic information is expressed by the symbol η_{ij}. This information indicates the ant's degree of interest and attention to explore new paths leading to a better fog node. In our proposed model, the calculation of heuristic information is done based on equation (5)

$$\eta_{ij} = \frac{1}{delay_{ij}(s_0)}. \tag{5}$$

Problem's defaults: The following points are considered as defaults to solve the problem:

- Initial assignment of tasks to fog nodes is done.
- Each ant receives a list of tasks and information on the fog nodes from the controller.
- Fixed parameters are based on the values given in the paper [58].
- The algorithm leads toward a near-optimal solution using iterative processes.
- The purpose of the problem is to reduce the response time by considering the delay time parameter.

2.8. Building a Solution (Motion Transfer Law)

The movement of the ant k to place task i on node j is similar to the Pseudo-Random proportional law (equation (6))

$$j = \begin{cases} max_{u \in \Omega_{k(i)}} \left\{ [\tau_{iu}{}^{\alpha}] \cdot [\eta_{iu}{}^{\beta}] \right\}, & if\ q \leq q_O \\ l, & otherwise \end{cases}. \tag{6}$$

In this equation, α and β are weighting coefficients or compatible parameters. These parameters determine the amount of pheromone in the path and the heuristic functions used, respectively. It affects the probability of choosing a particular path. L is a random variable obtained from the probability distribution of equations (7). q is a uniform random number in the range [0, 1] and q_O is a parameter between zero and one, which controls the balance between discovering routes traveled so far and searching for unmet routes. If q is more minor than q_O, this process is called exploitation, and we select the fog node in the set $\Omega_{k(i)}$ that has the highest value according to equation (6). If q is more significant than q_O, the fog node in the set $\Omega_{k(i)}$ is randomly selected using the law of probability distribution ${p_{ij}}^k$; i.e., the probability that the ant k puts task i on the fog node j. This process, called exploration, improves random search and reduces complexity. Exploitation helps ants quickly converge to a high-quality solution while, at the same time, exploration prevents stagnation through providing a more comprehensive search space

$${p_{ij}}^k = \begin{cases} \dfrac{{\tau_{ij}}^{\alpha} \cdot {\eta_{ij}}^{\beta}}{\sum_{u \in \Omega_{k(i)}} {\tau_{iu}}^{\alpha} \cdot {\eta_{iu}}^{\beta}}, & if\ j \in \Omega_{k(i)} \\ 0, & otherwise \end{cases}, \tag{7}$$

where τ_{ij} indicates the amount of the pheromones at points $(i,\ j)$. $\Omega_{k(i)}$ is a set of fog nodes that can accept i functions. The remaining nodes of ant k are based on node i to construct an acceptable answer. If $\alpha = 0$, the algorithm works greedily so that the selection of the next node does not take into account the number of pheromones; therefore, the selection of the nearest route takes precedence. If $\beta = 0$, the algorithm considers only the number of pheromones, regardless of the path length.

2.9. Updating the Pheromones

In the Ant-colony optimization algorithm, the pheromone needs to be updated. The number of pheromones can increase or decrease. The amount of pheromone increases with sedimentation and decreases with evaporation. The deposition of new pheromones is based on the fact that pheromone pathways indicate the information in some solutions, and the movement in these reasonable solutions is built following the other ants' solutions. However, pheromone evaporation is a practical implementation of forgetting that prevents the algorithm from over-converging to non-optimal regions, leading to the search for new and desirable regions in the search space. In our proposed algorithm, the pheromone upgrade process consists of two stages: local pheromone upgrade and global pheromone upgrade when task i is placed on fog node j, the decrease in pheromone level between task i and fog node j is applied by the local upgrade given in equations (8)

$$\tau_{ij}(t) = (1 - \rho_L)\, \tau_{ij}(t-1) + \rho_L \cdot \tau_{ij}^0. \tag{8}$$

In which τ_0 is the level of the initial pheromone and ρ_L is the local pheromone evaporation parameter ($0<\rho_L<1$). The global pheromone upgrade applies after the solution for all the ants for one repetition has been completed. Since all non-dominated solutions or the Pareto set are considered optimal or good solutions for optimization problems, we assume that all non-dominated solutions are the same and of high quality and that all dominated solutions must be eliminated. Therefore, the global update is applied to each solution s of the Pareto set using equation (9)

$$\tau_{ij}(t) = (1-\rho_g)\,\tau_{ij}(t-1) + \frac{\rho_g \cdot \lambda}{total\_delay}(s_0). \tag{9}$$

In which:

$$\lambda = \frac{N_{ant}}{t - N_{iter.s} + 1}. \tag{10}$$

In equation (10), ρ_g ($0<\rho_L<1$) is the pheromone evaporation parameter in the global update, and non-dominated global solutions, in the form of Pareto sets, are stored in an external set.

In equation (10), N_{ant} shows the number of ants, and $N_{iter.s}$ shows the number of repetitions in the solution of s in the external set. λ is the coefficient of adaptation, which helps control the pheromone's information in an external set over time. The global pheromone upgrade aims to increase the ants' learning.

2.10. Pseudo-code

Implementation performed in the following psuedo-code is shown in below.

To determine the fog node (in addition to equation (6)), the probability of the fog node productivity is considered. It is in the range of 30 to 80%.

Based on the flowchart shown in Fig. 1, the fixed values (list of tasks and information on the fog nodes) are received as input for the problem. Then, to determine the initial amount of pheromone, random assignment of tasks is performed on the fog nodes, and their latency is calculated and stored as the initial pheromone. It then enters the repeating loop, and the solution of each ant is made. Each ant makes the solution by sorting the list of tasks at their disposal according to the deadlines in ascending order. Then, remove the first task from the beginning of the list and find the appropriate fog node according to equation (6). The ants also check the productivity of the fog node over and over again. The local pheromone is upgraded based on equations (8), and the ant stores its solution in the optimal solutions. After all the ants make the solution in one repetition, the set of solutions made by the ants are examined, the best ones are left in the list, and the global pheromone upgrade is done based on equations (9). These steps are repeated to ensure that the best solutions remain in the *s_optimal* set. *s_optimal* represents all assignments or assignments that have the shortest response time for tasks.

Since fog calculations are always under investigation and fog nodes have unpredictable behaviors, the real test platform is not yet available to most researchers, and they use simulation and modeling tools to evaluate their work. Since the results of cases such as real-time scheduling can be different from simulation, it can be concluded that implementing the mechanisms discussed in actual experiments is still challenging. Any proposed model to prove its claim; requires the presentation of acceptable results and comparison of the model with other previous models. A data set appropriate to the problem must first be selected and then simulated or modeled inappropriate software to evaluate a model. By using the design proposed in this work, the load

Algorithm 1 The proposed model's algorithm

Input: fog_node_list, $job\_list[size, deadline, process]$
Output: $pare\_to\_set$, $s\_optimal$
set values of n_ant, n_itr, n_itrmax, pl, pg, q_O, α, and β
initialize primary pheromone based on random allocation ▷ initialization
while $n\_itr < n\_itrmax$ **do** ▷ iterative loop
  for ant_k **do** ▷ construct solution
    while job_list is not empty **do**
      sort job_list based on the $deadline$
      get j_i from job_list
      choose fog_node with eq. 6-3 to place j_i
      $flag\_underload \leftarrow 0$
      $flag\_overload \leftarrow 0$
      detect()
      if $fog\_node[utilization] > 0.8$ **then**
        $flag\_overload \leftarrow 1$
      else if $fog\_node[utilization] < 0.3$ **then**
        $flag\_underload \leftarrow 1$
      end if
      remove j_i from job_list
      apply local pheromone update based one eq 8-3
    end while
    add ant_k solution to $s\_optimal$
  end for
  for s_i in $s\_optimal$ **do** ▷ construct optimal solution
    apply local pheomone update based one eq 9-3
  end for
  $n\_itr \leftarrow n\_itr + 1$
end while

balance between the fog nodes can be created so that in addition to reducing the delay, the response time of the tasks is also reduced.

3. Evaluation

The load balancing problem in the fog environment is implemented based on the Ant-colony algorithm in Python. The reason for choosing Python for this implementation is to provide the libraries needed for implementation. Since Python is a dynamic language, it has advantages such as brevity in coding, no function limitation, and the ability to replace functions during the performance. Due to its explicit nature, Python, compared to other languages such as Java or C++, helps improve productivity and reduce development time.

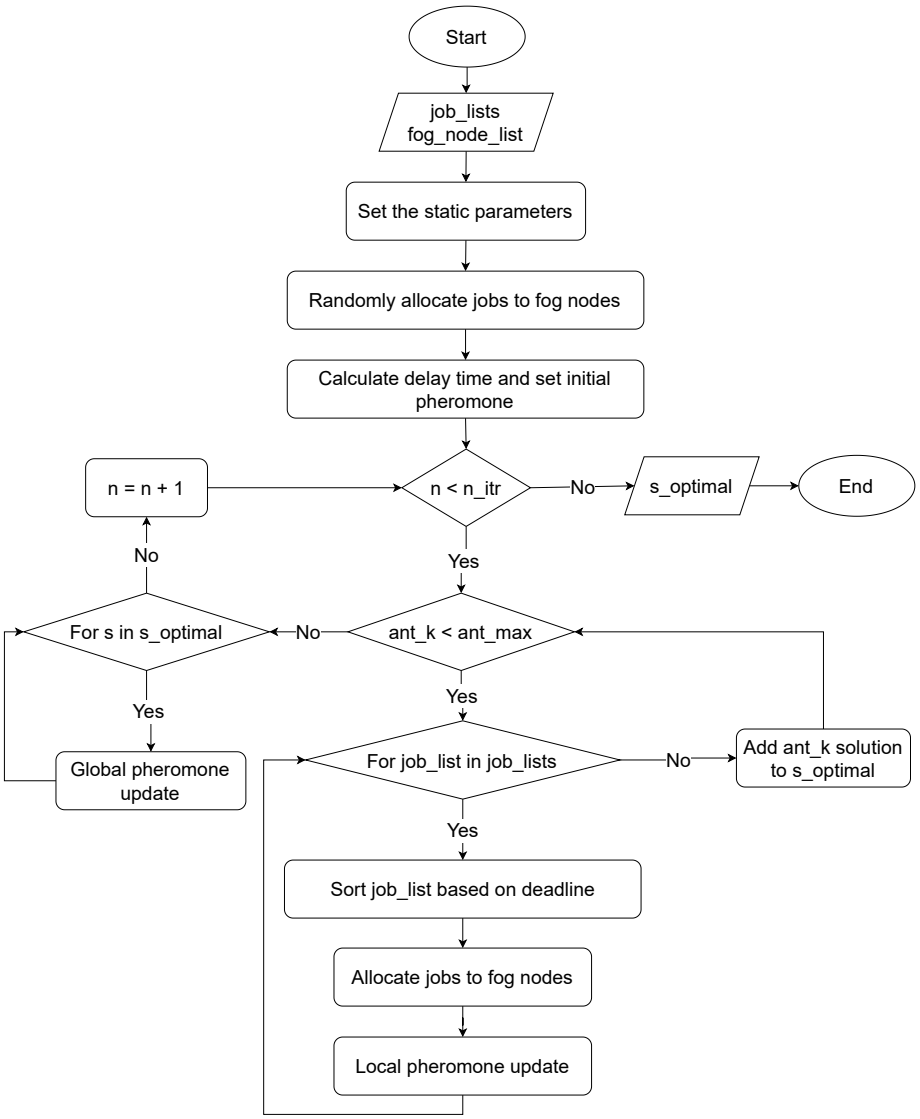

Figure 1. The proposed model's flowchart

3.1. Evaluation Data Set

Evaluation data sets can be examined in two categories: real and artificial data sets. Real data sets are data published by reputable companies used for evaluation. Artificial data sets are produced by different statistical models and are determined based on their parameters. In our proposed model, we use a random data set. In this set, the volume of tasks and their deadlines and processing power are generated and stored randomly. Also, to ensure the obtained results, 200 evaluation tests have been performed, and the announced results are the average of these 200 evaluations.

3.2. Evaluation Criteria

The objective function for our proposed model can be extended to a multi-objective function. Therefore, in this study, we focused on one objective. Our proposed model considers response time as the primary evaluation criterion [1] and shows that making a decision based on delay time can also reduce response time.

3.3. Evaluating the Proposed Model

Our proposed algorithm is implemented with Python programming language and evaluated on three servers. These servers had 24-core physical specifications with an E5-2650 v3 processor and a frequency of 2.3 GHz. Several virtual machines, including an 8-core virtual machine with 32 GB of random memory as the fog control node, 150 dual-core virtual machines with 4 GB of random memory as fog nodes, and 400 Mbps bandwidth are defined on these servers. Table 1 lists the features related to the tasks.

Table 1. Features related to the tasks Description

Description	Value	Parameter
The size of each task	10–150 MB	Task-size
Task processing power	500–2500 Cycle/s	Task-list

3.4. Evaluation Results

We evaluated our proposed model based on the algorithms given in the paper [45]. The criterion measured was the task response time. The following Tab. 2 to Tab. 5 shows the response times for evaluated models. This experiment was repeated with the number of tasks 400, 600, 1000.

Table 2. Table related to 400 tasks

	The Proposed model	The model in [45]
Average response time per task (milliseconds)	12	20
Run time (hours)	1	1.15

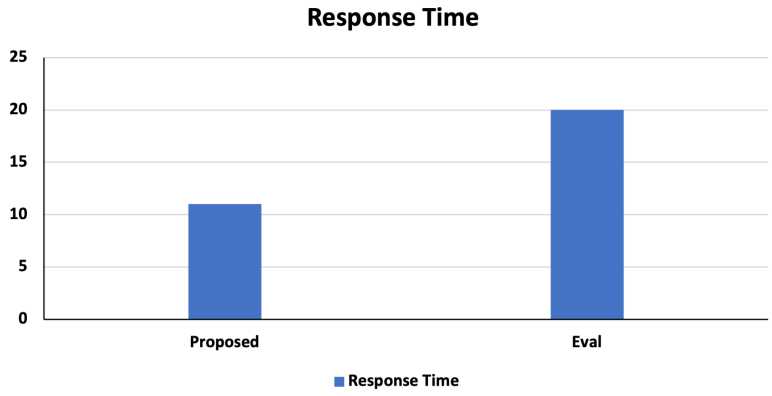

Figure 2. Response time for 400 tasks

It can be seen in the above tables and the resulting graphs that, as the number of tasks increases, the execution time to find the load balance between the fog nodes increases. Consequently, the average response time of tasks also increases as the load density on the fog nodes increases, and it becomes more challenging to find the correct node for each task over time. Table 5 shows the average latency based on the number of tasks.

Table 3. Table related to 600 tasks

	The Proposed Model	The model in [45]
Average response time per task (milliseconds)	20	25
Run time (hours)	0.06	0.1

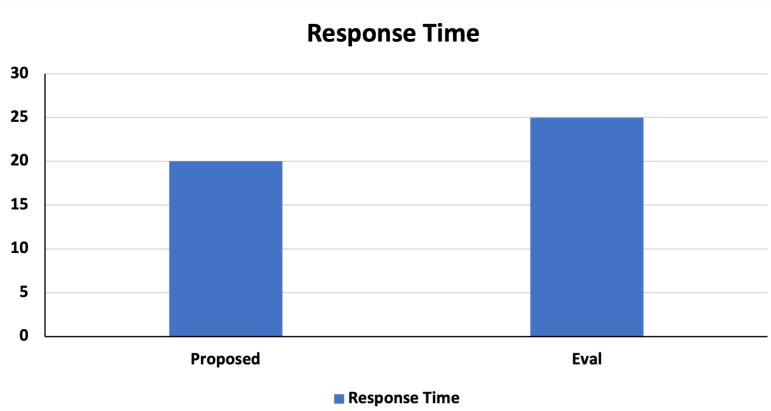

Figure 3. Response time for 600 tasks

Table 4. Table related to 1000 tasks

	The Proposed model	The model in [45]
Average response time per task (milliseconds)	27	38
Run time (hours)	0.13	6

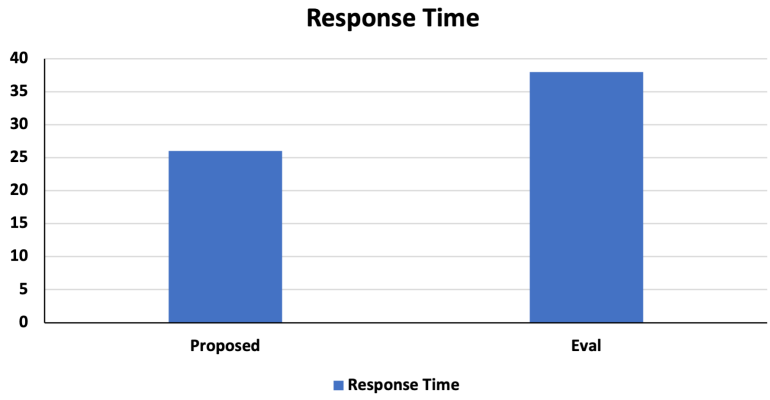

Figure 4. Response time for 1000 tasks

Conclusion

Due to the Internet of Things explosive expansion, processing speed improvements, and mobile technology, fog-based processing is crucial. This effective processing approach meets the requirements of delay-sensitive applications and eases the network bandwidth bottleneck. One of the key problems with this architecture is load balance, which prevents overloading (and hence lowers the system's efficiency). In order to effectively utilize the resources, decrease response time and latency, and so boost system efficiency, it is necessary to balance the load

Table 5. Average latency based on the number of tasks

Tasks	400	600	1000
Average latency of each task (milliseconds)	9.8	15.3	20.2

amongst the fog nodes using efficient algorithms. In the current work, we demonstrated how the Ant-colony optimization algorithm, along with the delay criterion and the addition of the load criterion for the fog environment, may be utilized to balance the load on the fog nodes and speed up task completion. The load on the fog nodes must be balanced in order to minimize the latency, but identifying the ideal host (load distribution on resources) for loads is an NP-hard task. Because it takes a while to obtain the precise solution, optimization procedures are employed. The Ant-colony algorithm makes a choice based on the latency criterion and the inclusion of the load criterion in the fog computing environment because meta-heuristic models offer better opportunities to create a compromise between the complexity of the search process and optimization. The optimization parameters and mapping load distribution parameters are then thoroughly described. Finally, we designed the load balance between the fog nodes in such a way that reaction time was also lowered in addition to lowering latency using the simulation and comparison with the algorithm described in the paper [45]. We also saw that when the number of tasks increased, it took longer to find a load balance between the fog nodes, which increased the tasks' average response times (due to the increase in density and volume of load on the fog nodes and therefore finding the correct node for each task getting more difficult).

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Abbasi, S.H., Javaid, N., Ashraf, M.H., *et al.*: Load stabilizing in fog computing environment using load balancing algorithm. In: Advances on Broadband and Wireless Computing, Communication and Applications. pp. 737–750. Springer (2018). https://doi.org/10.1007/978-3-030-02613-4_66

2. Aghdashi, A., Mirtaheri, S.L.: Novel dynamic load balancing algorithm for cloud-based big data analytics. The Journal of Supercomputing 78(3), 4131–4156 (2022). https://doi.org/10.1007/s11227-021-04024-8

3. Ahmed, A., Arkian, H., Battulga, D., *et al.*: Fog computing applications: Taxonomy and requirements. CoRR abs/1907.11621 (2019), https://arxiv.org/abs/1907.11621

4. Al-Qamash, A., Soliman, I., Abulibdeh, R., Saleh, M.: Cloud, fog, and edge computing: A software engineering perspective. In: 2018 International Conference on Computer and Applications (ICCA). pp. 276–284. IEEE (2018). https://doi.org/10.1109/COMAPP.2018.8460443

5. Alagarsamy, M., Sundarji, A., Arunachalapandi, A., Kalyanasundaram, K.: Cost-aware ant colony optimization based model for load balancing in cloud computing. The International Arab Journal of Information Technology 18(5), 719–729 (2021)

6. Alboaneen, D.A., Tianfield, H., Zhang, Y.: Metaheuristic approaches to virtual machine placement in cloud computing: a review. In: 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC). pp. 214–221. IEEE (2016). https://doi.org/10.1109/ISPDC.2016.37

7. Aruna, K., Pradeep, G.: Ant Colony Optimization-based Light Weight Container (ACO-LWC) Algorithm for Efficient Load Balancing. Intelligent Automation & Soft Computing 34(1), 205–219 (2022). https://doi.org/10.32604/iasc.2022.024317

8. Atlam, H.F., Walters, R.J., Wills, G.B.: Fog computing and the Internet of Things: A review. Big Data Cogn. Comput. 2(2), 10 (2018). https://doi.org/10.3390/bdcc2020010

9. Baek, J.y., Kaddoum, G., Garg, S., Kaur, K., Gravel, V.: Managing fog networks using reinforcement learning based load balancing algorithm. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC). pp. 1–7. IEEE (2019). https://doi.org/10.1109/WCNC.2019.8885745

10. Beaulah Soundarabai, P., Thriveni, J., Venugopal, K., Patnaik, L.: Comparative study on load balancing techniques in distributed systems. International Journal of Information Technology and Knowledge Management 6(1), 53–60 (2012)

11. Beraldi, R., Canali, C., Lancellotti, R., Mattia, G.P.: Distributed load balancing for heterogeneous fog computing infrastructures in smart cities. Pervasive and Mobile Computing 67, 101221 (2020). https://doi.org/10.1016/j.pmcj.2020.101221

12. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog Computing and Its Role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. pp. 13–16. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2342509.2342513

13. Bukhsh, R., Javaid, N., Ali Khan, Z., *et al.*: Towards fast response, reduced processing and balanced load in fog-based data-driven smart grid. Energies 11(12), 3345 (2018). https://doi.org/10.3390/en11123345

14. Cao, K., Liu, Y., Meng, G., Sun, Q.: An overview on edge computing research. IEEE Access 8, 85714–85728 (2020). https://doi.org/10.1109/ACCESS.2020.2991734

15. Chandak, A., Ray, N.K.: A review of load balancing in fog computing. In: International Conference on Information Technology (ICIT). pp. 460–465. IEEE (2019). https://doi.org/10.1109/ICIT48102.2019.00087

16. Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: Handbook of Metaheuristics. pp. 250–285. Springer (2003). https://doi.org/10.1007/0-306-48056-5_9

17. Fan, Q., Ansari, N.: Towards workload balancing in fog computing empowered IoT. IEEE Transactions on Network Science and Engineering 7(1), 253–262 (2018). https://doi.org/10.1109/TNSE.2018.2852762

18. Ghomi, E.J., Rahmani, A.M., Qader, N.N.: Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications 88, 50–71 (2017). https://doi.org/10.1016/j.jnca.2017.04.007

19. Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 13(5), 533–549 (1986). https://doi.org/10.1016/0305-0548(86)90048-1, applications of Integer Programming

20. Gu, J., Mo, J., Li, B., Zhang, Y., Wang, W.: A multi-objective fog computing task scheduling strategy based on ant colony algorithm. In: 2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE). pp. 12–16. IEEE (2021). https://doi.org/10.1109/ICISCAE52414.2021.9590674

21. Hamrioui, S., Lorenz, P.: Load balancing algorithm for efficient and reliable IoT communications within E-health environment. In: GLOBECOM 2017-2017 IEEE Global Communications Conference. pp. 1–6. IEEE (2017). https://doi.org/10.1109/GLOCOM.2017.8254435

22. He, X., Ren, Z., Shi, C., Fang, J.: A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles. China Communications 13(Supplement2), 140–149 (2016). https://doi.org/10.1109/CC.2016.7833468

23. Ivanisenko, I.N., Radivilova, T.A.: Survey of major load balancing algorithms in distributed system. In: 2015 Information Technologies in Innovation Business Conference (ITIB). pp. 89–92. IEEE (2015). https://doi.org/10.1109/ITIB.2015.7355061

24. Jijin, J., Seet, B.C., Chong, P.H.J., Jarrah, H.: Service load balancing in fog-based 5G radio access networks. In: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). pp. 1–5. IEEE (2017). https://doi.org/10.1109/PIMRC.2017.8292300

25. Kamal, M.B., Javaid, N., Naqvi, S.A.A., *et al.*: Heuristic min-conflicts optimizing technique for load balancing on fog computing. In: Advances in Intelligent Networking and Collaborative Systems. pp. 207–219. Springer (2018). https://doi.org/10.1007/978-3-319-98557-2_19

26. Kaur, S., Kaur, G.: A review of load balancing strategies for distributed systems. International Journal of Computer Applications 121(18), 45–47 (2015). https://doi.org/10.5120/21644-4985

27. Khaneghah, E.M., Mirtaheri, S.L., Grandinetti, L., *et al.*: A dynamic replication mechanism to reduce response-time of I/O operations in high performance computing clusters. In: 2013 International Conference on Social Computing. pp. 738–743. IEEE (2013). https://doi.org/10.1109/SocialCom.2013.110

28. Khattak, H.A., Arshad, H., Ahmed, G., *et al.*: Utilization and load balancing in fog servers for health applications. EURASIP Journal on Wireless Communications and Networking 2019(1), 1–12 (2019). https://doi.org/10.1186/s13638-019-1395-3

29. Kishor, A., Chakarbarty, C.: Task offloading in fog computing for using smart ant colony optimization. Wireless Personal Communications 127, 1–22 (2021). https://doi.org/10.1007/s11277-021-08714-7

30. Kumar, A., Pandey, S., Prakash, V.: A survey: Load balancing algorithm in cloud computing. In: Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) (2019). https://doi.org/10.2139/ssrn.3350328

31. Mahmud, R., Kotagiri, R., Buyya, R.: Fog computing: A taxonomy, survey and future directions. Internet of Everything pp. 103–130 (2018). https://doi.org/10.1007/978-981-10-5861-5_5

32. Manju, A., Sumathy, S.: Efficient load balancing algorithm for task preprocessing in fog computing environment. In: Smart Intelligent Computing and Applications. pp. 291–298. Springer (2019). https://doi.org/10.1007/978-981-13-1927-3_31

33. Mirtaheri, S.L., Fatemi, S.A., Grandinetti, L.: Adaptive load balancing dashboard in dynamic distributed systems. Supercomputing Frontiers and Innovations 4(4), 34–49 (2017). https://doi.org/10.14529/jsfi170403

34. Mirtaheri, S.L., Grandinetti, L.: Optimized load balancing in high-performance computing for big data analytics. Concurrency and Computation: Practice and Experience 33(16), e6265 (2021). https://doi.org/10.1002/cpe.6265

35. Naqvi, S.A.A., Javaid, N., Butt, H., *et al.*: Metaheuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid. In: Advances in Network-Based Information Systems. pp. 700–711. Springer (2019). https://doi.org/10.1007/978-3-319-98530-5_61

36. Neto, E.C.P., Callou, G., Aires, F.: An algorithm to optimise the load distribution of fog environments. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 1292–1297. IEEE (2017). https://doi.org/10.1109/SMC.2017.8122791

37. Ningning, S., Chao, G., Xingshuo, A., Qiang, Z.: Fog computing dynamic load balancing mechanism based on graph repartitioning. China Communications 13(3), 156–164 (2016). https://doi.org/10.1109/CC.2016.7445510

38. Patel, N., Chauhan, S.: A survey on load balancing and scheduling in cloud computing. International Journal for Innovative Research in Science and Technology 1(7), 185–189 (2015)

39. Puthal, D., Obaidat, M.S., Nanda, P., *et al.*: Secure and sustainable load balancing of edge data centers in fog computing. IEEE Communications Magazine 56(5), 60–65 (2018). https://doi.org/10.1109/MCOM.2018.1700795

40. Qun, R., Arefzadeh, S.M.: A new energy-aware method for load balance managing in the fog-based vehicular ad hoc networks (VANET) using a hybrid optimization algorithm. IET Communications 15(13), 1665–1676 (2021). https://doi.org/10.1049/cmu2.12179

41. Rathod, D., Chowdhary, G.: Load balancing of fog computing centers: minimizing response time of high priority requests. International Journal of Innovative Technology and Exploring Engineering 8(11), 2713–2716 (2019)

42. Shah, J.M., Kotecha, K., Pandya, S., *et al.*: Load balancing in cloud computing: Methodological survey on different types of algorithm. In: 2017 International Conference on Trends in Electronics and Informatics (ICEI). pp. 100–107. IEEE (2017). https://doi.org/10.1109/ICOEI.2017.8300865

43. Shams, P., Mirtaheri, S.L., Shahbazian, R., Arianyan, E.: Improving IoT resource management using fog calculations and ant lion optimization algorithm. Journal of Information and Communication Technology 55(56), 1–19 (2023)

44. Sharma, S., Singh, S., Sharma, M.: Performance analysis of load balancing algorithms. World academy of science, engineering and technology 38(3), 269–272 (2008)

45. Sharma, S., Saini, H.: Efficient solution for load balancing in fog computing utilizing artificial bee colony. International Journal of Ambient Computing and Intelligence (IJACI) 10(4), 60–77 (2019). https://doi.org/10.4018/IJACI.2019100104

46. Sharma, S., Saini, H.: A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. Sustainable Computing: Informatics and Systems 24, 100355 (2019). https://doi.org/10.1016/j.suscom.2019.100355

47. Sthapit, S., Hopgood, J.R., Thompson, J.: Distributed computational load balancing for real-time applications. In: 2017 25th European Signal Processing Conference (EUSIPCO). pp. 1385–1189. IEEE (2017). https://doi.org/10.23919/EUSIPCO.2017.8081436

48. Talaat, F.M., Ali, S.H., Saleh, A.I., Ali, H.A.: Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks. Journal of Network and Systems Management 27(4), 883–929 (2019). https://doi.org/10.1007/s10922-019-09490-3

49. Téllez, N., Jimeno, M., Salazar, A., Nino-Ruiz, E.: A tabu search method for load balancing in fog computing. Int. J. Artif. Intell 16(2), 1–30 (2018)

50. Verma, M., Bhardwaj, N., Yadav, A.K.: Real time efficient scheduling algorithm for load balancing in fog computing environment. Int. J. Inf. Technol. Comput. Sci 8(4), 1–10 (2016). https://doi.org/10.5815/ijitcs.2016.04.01

51. Verma, M., Yadav, N.B.A.K.: An architecture for load balancing techniques for Fog computing environment. International Journal of Computer Science and Communication 8(2), 43–49 (2015)

52. Verma, S., Yadav, A.K., Motwani, D., *et al.*: An efficient data replication and load balancing technique for fog computing environment. In: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). pp. 2888–2895. IEEE (2016)

53. Wan, J., Chen, B., Wang, S., *et al.*: Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Transactions on Industrial Informatics 14(10), 4548–4556 (2018). https://doi.org/10.1109/TII.2018.2818932

54. Wang, J., Li, D.: Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. Sensors 19(5), 1023 (2019). https://doi.org/10.3390/s19051023

55. Xu, X., Fu, S., Cai, Q., *et al.*: Dynamic resource allocation for load balancing in fog environment. Wireless Communications and Mobile Computing 2018 (2018). https://doi.org/10.1155/2018/6421607

56. Yi, S., Hao, Z., Qin, Z., Li, Q.: Fog computing: Platform and applications. In: 2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb). pp. 73–78. IEEE (2015). https://doi.org/10.1109/HotWeb.2015.22

57. Zahid, M., Javaid, N., Ansar, K., *et al.*: Hill climbing load balancing algorithm on fog computing. In: Advances on P2P, Parallel, Grid, Cloud and Internet Computing. pp. 238–251. Springer (2018). https://doi.org/10.1007/978-3-030-02607-3_22

58. Zhao, H., Wang, J., Liu, F., *et al.*: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Transactions on Parallel and Distributed Systems 29(6), 1385–1400 (2018). https://doi.org/10.1109/TPDS.2018.2794369