A Load Balancing Algorithm for 5G Vehicular Cloud Computing Systems

Eirini Zoumi
Dept. of Informatics, University of Piraeus
Piraeus, Greece
zoumi@unipi.gr

Emmanouil Skondras
Dept. of Informatics, University of Piraeus
Piraeus, Greece
skondras@unipi.gr

David Veliu
Dept. of Informatics, University of Western Macedonia
Kastoria, Greece
davidveliu1994@gmail.com

Angelos Michalas
Dept. of Electrical and Computer Engineering, University of Western Macedonia
Kozani, Greece
amichalas@uowm.gr

Dimitrios D. Vergados
Dept. of Informatics, University of Piraeus
Piraeus, Greece
vergados@unipi.gr

ABSTRACT

5G Vehicular Cloud Computing (5G-VCC) infrastructures are evolving rapidly. In a 5G-VCC system, Cloud resources should be efficiently distributed to provide satisfactory Quality of Service (QoS) for modern services with increased requirements. Furthermore, the workload should be fairly distributed to the available Virtual Machines (VMs). This paper proposes an algorithm for performing load balancing in Cloud infrastructures that exist in 5G-VCC systems. The algorithm is called Modified Ant Colony Optimization (MACO), as its functionality is influenced by the natural behaviour of ants. Specifically, the MACO algorithm assigns a pheromone (weight) value to each VM. Subsequently, for each service request the VM with the highest pheromone is selected in a way similar to the one that ants apply to select optimal routes. The selection of each VM results in the decrement its pheromone value, considering the workload of the assigned service. Evaluation results show that the proposed algorithm outperforms existing load balancing algorithms in terms of the processing time required for serving the user requests.

CCS CONCEPTS

• **Networks** → **Network architectures**; • **Computer systems organization** → **Real-time system architecture**; • **Computing methodologies**;

KEYWORDS

5G-VCC systems, cloud services, Ant Colony Optimization algorithm, load balancing

ACM Reference Format:
Eirini Zoumi, Emmanouil Skondras, David Veliu, Angelos Michalas, and Dimitrios D. Vergados. 2020. A Load Balancing Algorithm for 5G Vehicular Cloud Computing Systems. In *PCI '20: 24th Pan-Hellenic Conference on Informatics, November 20–22, 2020, Athens, GR.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3437120.3437316

1 INTRODUCTION

Fifth Generation Vehicular Cloud Computing (5G-VCC) [12] combines the operating principles of both Vehicular Networks [5] and Cloud computing [9]. Network access technologies, such as 3GPP Long Term Evolution Advanced Pro with Full-Dimension Multiple Input Multiple Output (LTE-A Pro FD-MIMO) [8] are used for the interaction between the vehicles and the Cloud infrastructure.Vehicles interact with a Cloud infrastructure, which offers a variety of modern services with strict Quality of Service (QoS) constraints.

The computational and storage resources of the Cloud should be distributed optimally to user services, while at the same time their utilization should be maximized. Load balancing [7] aims to improve the satisfaction of the requirements of user services as well as to increase the utilization of the Cloud resources. By applying an effective load balancing algorithm, both the extensibility and the availability of the system are improved. Specifically, extensibility is the ability of the algorithm to give a simplified solution even if either the number of users or the number of available resources increases unexpectedly. Furthermore, the availability factor is referred to the case when the system remains accessible even in cases where the incoming workload requires increased system resources, or failures occur in a part of the systems resources. Indicatively, if operational failures occur to a serving VM, then the user requests are assigned to other VMs.

In this paper, the Modified Ant Colony Optimization (MACO) load balancing algorithm is proposed. The functionality of MACO is influenced by the natural behaviour of ants, while centralized control is performed by applying the operating principles of the Software Defined Networks (SDN) [3]. For each user service, the MACO algorithm assigns a pheromone value to each available

VM, in a way similar to the one that ants apply to select optimal routes. Subsequently, the VM with the higher pheromone is selected to serve a user service. Additionally, it has to be noted that Interval Valued Trapezoidal Fuzzy Numbers (IVTFNs) [14] are used for the representation of the pheromone values. The proposed algorithm is compared with existing ones considering multiple scenarios.

The remainder of the paper is organized as follows: Section 2 overviews the existing related work, Section 3 describes the proposed algorithm, Section 4 presents the evaluation results and, finally, Section 5 concludes the described work.

2 RELATED WORK

Several algorithms have been proposed in the research literature for performing load balancing.

Indicatively, ACO [16] algorithm is inspired from ant search mechanisms. An advantage of the algorithm is that it can look for solutions with parallel mechanisms. The function of ACO is influenced by the natural behavior of ants. Ants, during the process of finding food, manage to find the shortest path from the food to their nest. This is achieved because the ants as they move leave behind a pheromone, which can be detected by others. Depending on the distance and quality of the food, the pheromone becomes more intense or weakens. ACO behaves in a similar way. Initially it tries many solutions and as it compares them with each other, it adds pheromone to them depending on the effectiveness until the best solution is found.

In [7] the Round Robin, the Weighted Round Robin and the Randomized algorithms are described. Specifically, the Round Robin algorithm distributes sequentially the user requests to a list of virtual machines. When a new request is assigned to the last virtual machine of the list, the assignment of the next requests will start again from the from the beginning of the list, namely from the first virtual machine. It should be noted that the Weighted Round Robin algorithm differs from the Round Robin, giving weight to each virtual machine depending on its capabilities. Thus, the selection of the Round Robin is improved, since the virtual machine with the highest weight, namely with the highest technical specifications, will serve a higher load of requests. Furthermore, the Randomized algorithm randomly selects a virtual machine for serving each new request.

In [6] the Throttled algorithm is proposed. It maintains a list of virtual machines along with their availability status and their workload. When a new user request arrives, the algorithm finds an available virtual machine and assigns to it the request. If the virtual machine becomes unavailable, the algorithm looks for the next available virtual machine. If none of the virtual machines are available, then the algorithm maintains the request in a waiting list until a machine is released.

The Response Time and Weighted Response Time [7] algorithm is based on the response time of the virtual machines. It distributes requests to achieve a shorter response time. The algorithm uses two ways of monitoring, namely the internal and the external monitoring. The internal monitoring uses the movement of data between the system and the user to detect the response time. On the other hand, in external monitoring, the algorithm assigns a request specifically to a virtual machine to measure the response time. Response time should be measured frequently to make the distribution of work more efficient and preferably the most recent measurement should be used. In the weighted response time algorithm, the information about the times comes from the status checks of the machines and the weight must be in the most recent measurement.

Honeybee Foraging Algorithm [10] comes from the behavior of bees. There are two types of bees, namely the detectors and the food collectors. Detectors leave the nest and try to find food sources. After locating the source, they return to the nest and present the quantity and quality or the distance. Then the collectors come out and collect the honey from the springs. Having collected the honey and returned to the nest then they show how much is left. A request in the Cloud can be done through the average processing checks that a request will need and the average performance check of each node in the system. The algorithm with appropriate calculations creates criteria and tries to find the request that will best suit one of the machines on the system. When a machine is found to be overloaded then the algorithm transfers the processes to another one.

3 THE PROPOSED LOAD BALANCING ALGORITHM

The proposed algorithm improves the Ant Colony Optimization (ACO) [16] methodology, whose function is influenced by the natural behavior of ants. It is called Modified ACO (MACO) and uses a weighted value, which is called pheromone, in order to evaluate each Virtual Machine (VM) of a Cloud infrastructure.

Specifically, there are N VMs in the Cloud environment used for the execution of the services, denoted as $V = v_1, v_2, \ldots, v_N$. An initial pheromone value $\hat{p}$ is assigned to each VM indicating the capabilities of its resources. The values of the VMs' pheromones are represented using Interval Valued Trapezoidal Fuzzy Numbers (IVTFNs) [4]. In particular, an Interval-valued fuzzy numbers (IVFN) is defined as $A = [A^L, A^U]$ and consists of the lower A^L and the upper A^U fuzzy numbers. IVFNs replace the crisp membership values with intervals in $[0, 1]$. They were proposed due to the fact that fuzzy information can be better expressed by intervals than by single values. The IVTFNs which are a generalized version of the IVFNs can be represented as: $A = [A^L, A^U] = [(x_1^L, x_2^L, x_3^L, x_4^L, v_{A^L}), (x_1^U, x_2^U, x_3^U, x_4^U, v_{A^U})]$ where: $0 \leq x_1^L \leq x_2^L \leq x_3^L \leq x_4^L \leq 1$, $0 \leq x_1^U \leq x_2^U \leq x_3^U \leq x_4^U \leq 1$, $0 \leq v_{A^L} \leq v_{A^U} \leq 1$ and $A^L \subset A^U$. The operational rules of the IVTFNs are defined in [14] Thus, the algorithm produces a list of pheromones that describe the performance of each VM. In this way, the VMs are listed by priority, since VMs with better resources obtain higher pheromone values.

Subsequently, there are M users that request services of the Cloud infrastructure, denoted as $U = u_1, u_2, \ldots, u_M$. The algorithm assigns each user u to the VM with the higher pheromone. After each assignment, the pheromone of the corresponding VM is updated using formula ?? where the resource utilization $\hat{r}^s_{utilization}$ that each service s causes to its VM is considered. It has to be noted that the resource utilization factor is also represented using IVTFNs, whilet the $\ominus$ indicates the substitution operator of two fuzzy numbers as defined in [15]. As a result, a VM that serve the most demanding services obtain the higher decrements to its pheromone.

$$\begin{aligned}\hat{p}_{new} &= \hat{p}_{prev} \ominus \hat{r}^{s}_{utilization} \\ &= \Big[\Big(p^{L}_{prev,1} - r^{s,L}_{utilization,1}, p^{L}_{prev,2} - r^{s,L}_{utilization,2}, p^{L}_{prev,3} \\ &\quad -r^{s,L}_{utilization,3}, p^{L}_{prev,4} - r^{s,L}_{utilization,4}, v_{A^L}\Big), \\ &\quad \Big(p^{U}_{prev,1} - r^{s,U}_{utilization,1}, p^{U}_{prev,2} - r^{s,U}_{utilization,2}, p^{U}_{prev,3} \\ &\quad -r^{s,U}_{utilization,3}, p^{U}_{prev,4} - r^{s,U}_{utilization,4}, v_{A^U}\Big)\Big]\end{aligned}$$

Algorithm 1 presents the proposed methodology.

Algorithm 1 The pseudocode of the MACO algorithm.

```
Set $V = \{v_1, v_2, \ldots, v_N\}$ the set of available VMs ,
Set $U = \{u_1, u_2, \ldots, u_M\}$ the set of users requiring Cloud services
for each v ∈ V do
    Set initial pheromone to v
end for
for each user u ∈ U do
    Assign u to v with max pheromone
    Update v pheromone using formula (1)
end for
```

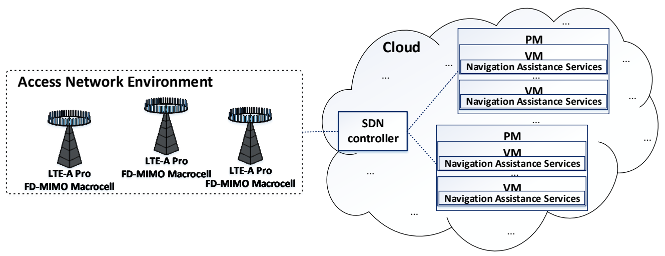
Cloud
Access Network Environment
LTE-A Pro FD-MIMO Macrocell
LTE-A Pro FD-MIMO Macrocell
LTE-A Pro FD-MIMO Macrocell
SDN controller
PM
VM
Navigation Assistance Services
VM
Navigation Assistance Services
PM
VM
Navigation Assistance Services
VM
Navigation Assistance Services

Figure 1: The simulated topology.

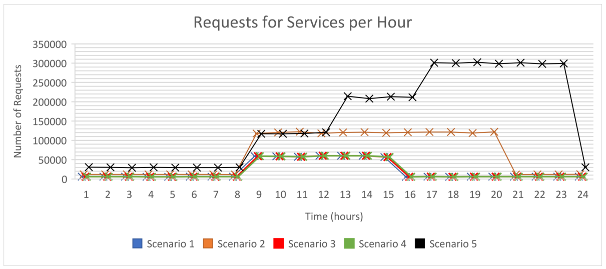

Figure 2: The number of user requests for obtaing access to the Navigation Assistance services.

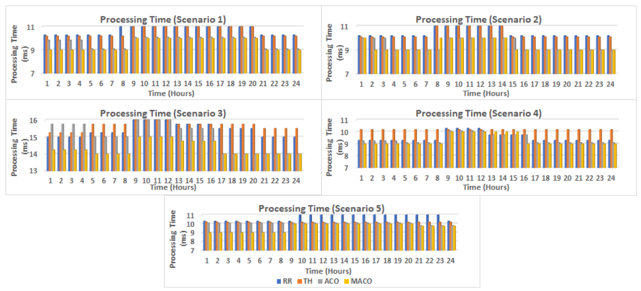

Figure 3: The proccessing times.

4 SIMULATION SETUP AND EVALUATION RESULTS

In our experiments, the 5G wireless network architecture presented in figure 1 is considered. It includes a Cloud infrastructure and an access network infrastructure.

The Cloud infrastructure has been implemented using the Cloud-Analyst [11] module of the CloudSim [1] platform. It includes a set of Virtual Machines (VMs) providing Navigation Assistance [13, 17] services. Specifically, in a Navigation Assistance service, information from the vehicles, the pedestrians and the city environment can be collected in order to assess the navigation of both vehicles and pedestrian users considering situations such as the current road traffic or road accidents. Indicatively, the Cloud infrastructure processes the entire mobility information received from all vehicles and extracts traffic information about a specific geographic area. Subsequently, the Cloud infrastructure informs all the vehicles that travel in the specified area about the traffic conditions. Furthermore, a Navigation Assistance orchestrates the available traffic lights in order to tackle issues such as traffic congestion.

The access network infrastructure has been implemented using the Network Simulator 3 (NS-3) [2] simulator. It consists of 4 LTE-A FD-MIMO Macrocells providing access to the Cloud services. Furthermore, a Software Defined Network (SDN) controller provides centralized control of the entire system. The linguistic terms for the determination of the performance of each VM as well as for the determination of the resource utilization that each service causes to its VM are represented by IVTFNs as shown in table 1.

The performance of the MACO algorithm is compared with the one observed from the Round Robin [7], the Throttled [6] and the ACO [16] algorithms, considering five different scenarios. Figure 2 presents the number of user requests for obtaining access to the Navigation Assistance services performed per hour according to each scenario, for a 24-hour period. As it can be observed, scenarios 2, 3 and 4 determine similar number of requests per hour. Also, the scenario 5 determines higher number of requests per hour, while scenario 1 determines intermediate number of requests per hour.

During the first scenario, the Cloud includes 10 VMs offering Good (G) performance each. Subsequently, during the second scenario the Cloud infrastructure includes 20 VMs offering Good (G) performance each. Additionally, the third, the fourth and the fifth scenarios include 50 VMs each. Specifically, in case of the third scenario each VM offers Medium (M) performance, while in cases of the fourth and the fifth scenario each VM offers Absolutely Good (AG) performance. Figure 3 presents the results of the considered algorithms with respect to the numbers of service requests performed in each scenario. As it can be observed the MACO algorithm outperforms the other ones in all cases, in terms of the processing time required for serving the number of requests per hour for the 24-hour period.

In particular, during the first scenario the algorithm with the slowest response is Round Robin, followed by Throttled. ACO algorithms perform better than the other two, while MACO succeeds lower response times from the other algorithms. Subsequently, during the second scenario the order of performance of the algorithms remains the same as in the first scenario, while at the same time an increase in time differences between algorithms is noticed, since

Table 1: Linguistic terms and the corresponding interval-valued trapezoidal fuzzy numbers used for the determination of the performance of each VM as well as of the resource utilization that each service causes .

Ling. term for the performance of each VM	Interval-valued trapezoidal fuzzy number	Ling. term for the resource utilization	Interval-valued trapezoidal fuzzy number
Absolutely Poor (AP)	[(0.0, 0.0, 0.0, 0.0, 0.8), (0.0, 0.0, 0.0, 0.0, 1)]	Absolute Low (AL)	[(0.0, 0.0, 0.062, 0.093, 0.8), (0.0, 0.0, 0.083, 0.125, 1)]
Very Poor (VP)	[(0.01, 0.02, 0.03, 0.07, 0.8), (0.0, 0.01, 0.05, 0.08, 1)]	Very Low(VL)	[(0.072, 0.104, 0.229, 0.26, 0.8), (0.041, 0.083, 0.25, 0.291, 1)]
Poor (P)	[(0.04, 0.1, 0.18, 0.23, 0.8), (0.02, 0.08, 0.2, 0.25, 1)]	Medium Low (ML)	[(0.239, 0.27, 0.395, 0.427, 0.8), (0.208, 0.25, 0.416, 0.458, 1)]
Medium Poor (MP)	[(0.17, 0.22, 0.36, 0.42, 0.8), (0.14, 0.18, 0.38, 0.45, 1)]	Medium (M)	[(0.406, 0.437, 0.562, 0.593, 0.8), (0.375, 0.416, 0.583, 0.625, 1)]
Medium (M)	[(0.32, 0.41, 0.58, 0.65, 0.8), (0.28, 0.38, 0.6, 0.7, 1)]	Medium High (MH)	[(0.572, 0.604, 0.729, 0.76, 0.8), (0.541, 0.583, 0.75, 0.791, 1)]
Medium Good (MG)	[(0.58, 0.63, 0.8, 0.86, 0.8), (0.5, 0.6, 0.9, 0.92, 1)]	Very High (VH)	[(0.739, 0.77, 0.895, 0.927, 0.8), (0.708, 0.75, 0.916, 0.958, 1)]
Good (G)	[(0.72, 0.78, 0.92, 0.97, 0.8), (0.7, 0.75, 0.95, 0.98, 1)]	Absolute High (AH)	[(0.906, 0.937, 1.0, 1.0, 0.8), (0.875, 0.916, 1, 1, 1)]
Very Good (VG)	[(0.93, 0.98, 1, 1, 0.8), (0.9, 0.95, 1, 1, 1)]		
Absolutely Good (AG)	[(1, 1, 1, 1, 0.8), (1, 1, 1, 1, 1)]		

more VMs (and thus more resources) are available. Furthermore, in the third scenario the MACO algorithm also outperforms the other three techniques. In this case, the ACO in the early stages of the scenario has increased delays performance to the Round Robin and Throttled and then approaches the MACO's performance. On the other hand, in the fourth scenario the response times have been reduced for all the algorithms since each VM offers increased performance. In this case, the Round Robin performs better than Throttled and the ACO algorithms. Round Robin's performance increase is likely to be due to the fact that increasing the number of the machines maintains fewer virtual ones, while reduces the total number of requests for each VM and PM. As a result, the load on each machine is reduced, and so is their process. As Throttled and ACO algorithms assign services in a waiting list if there is no waiting machine, the services are not processed at the same time, so in this scenario the system performs better. However, the MACO succeeds better results from the other algorithms. Finally, during the fifth scenario the ordering of the considered algorithms remains the similar with the previous scenarios with the MACO algorithm succeeding the better performance.

5 CONCLUSION

In this paper, the Modified Ant Colony Optimization (MACO) described. The proposed algorithm applied to a 5G-VCC system architecture. Its performance evaluated in comparison with the ACO, the Throttled and the Round Robin algorithms considering five different scenarios. According to the simulation results the proposed algorithm outperforms the existing solutions in terms of time required for requests about Navigation Assistance services. Future work includes the evaluation of the proposed algorithm in a real-time environment where the data is changing, as user requests will vary both in size and in functions that need to be performed by the system to process them. Finally, it is necessary to develop functions that predict and manage errors that may occur in a real environment, such as failures that can occurr during the VMs operation.

ACKNOWLEDGMENTS

The publication of this paper has been partly supported by the University of Piraeus Research Center (UPRC).

REFERENCES

[1] [n.d.]. CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. http://cloudbus.org/cloudsim/. Accessed: 2020.

[2] [n.d.]. Network Simulator 3 (NS3). https://www.nsnam.org/. Accessed: 2020.

[3] Jacob H Cox, Joaquin Chung, Sean Donovan, Jared Ivey, Russell J Clark, George Riley, and Henry L Owen. 2017. Advancing software-defined networks: A survey. *IEEE Access* 5 (2017), 25487–25526.

[4] Ali Ebrahimnejad. 2016. Fuzzy linear programming approach for solving transportation problems with interval-valued trapezoidal fuzzy numbers. *Sādhanā* 41, 3 (2016), 299–316.

[5] Xiaohu Ge. 2019. Ultra-reliable low-latency communications in autonomous vehicular networks. *IEEE Transactions on Vehicular Technology* 68, 5 (2019), 5005–5016.

[6] Saurabh Gupta, Amit Dixit, and Harsh Dev. 2017. A study on various load balancing algorithms for response time reduction in cloud environment. *International Journal of Current Engineering and Scientific Research (IJCESR)* (2017).

[7] Siham Hamadah. 2017. A survey: a comprehensive study of static, dynamic and hybrid load balancing algorithms. *International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN* (2017), 2249–9555.

[8] Hyoungju Ji, Younsun Kim, Juho Lee, Eko Onggosanusi, Younghan Nam, Jianzhong Zhang, Byungju Lee, and Byonghyo Shim. 2016. Overview of full-dimension MIMO in LTE-advanced pro. *IEEE Communications Magazine* 55, 2 (2016), 176–184.

[9] Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida. 2020. Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences, Elsevier* 32, 2 (2020), 149–158.

[10] Monika Rathore, Sarvesh Rai, and Navdeep Saluja. 2016. Randomized Honey Bee Load Balancing Algorithm in Cloud Computing System. *International Journal of Computer Science and Information Technologies* 7, 2 (2016), 703–707.

[11] Faizan Saeed, Nadeem Javaid, Muhammad Zubair, Muhammad Ismail, Muhammad Zakria, Muhammad Hassaan Ashraf, and Muhammad Babar Kamal. 2018. Load Balancing on Cloud Analyst Using First Come First Serve Scheduling Algorithm. (2018), 463–472.

[12] Emmanouil Skondras, Angelos Michalas, and Dimitrios D Vergados. 2019. Mobility management on 5g vehicular cloud computing systems. *Vehicular Communications, Elsevier* 16 (2019), 15–44.

[13] Adam J Spiers and Aaron M Dollar. 2016. Design and evaluation of shape-changing haptic interfaces for pedestrian navigation assistance. *IEEE transactions on haptics* 10, 1 (2016), 17–28.

[14] Shih-Hua Wei and Shyi-Ming Chen. 2009. Fuzzy risk analysis based on interval-valued fuzzy numbers. *Expert Systems with Applications* 36, 2 (2009), 2285–2299.

[15] Tong Wu, Xin-Wang Liu, and Shu-Li Liu. 2015. A fuzzy ANP with interval type-2 fuzzy sets approach to evaluate enterprise technological innovation ability. (2015), 1–8.

[16] Peng Xu, Guimin He, Zhenhao Li, and Zhongbao Zhang. 2018. An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization. *International Journal of Distributed Sensor Networks* 14, 12 (2018), 1550147718793799.

[17] Jyun-Yan Yang, Li-Der Chou, and Yao-Jen Chang. 2015. Electric-vehicle navigation system based on power consumption. *IEEE Transactions on Vehicular Technology* 65, 8 (2015), 5930–5943.