Placement of Edge Servers in Mobile Cloud Computing using Artificial Bee Colony Algorithm

Bing Zhou*, Bei Lu, Zhigang Zhang
College of Artificial Intelligence, Jiaozuo University, Jiaozuo, Henan, 454000, China

***Abstract*—Utilizing smart mobile devices for entertainment, education, and social networking has grown recently. Even though mobile applications are getting more sophisticated and resource-intensive, the computing power of mobile devices is still constrained. Mobile phone applications can perform better by shifting parts of their functions to cloud servers. However, because the cloud is frequently positioned far from mobile phone users, there may be a significant and unpredictable delay in the data transfer between users and the cloud. It is crucial for mobile applications since customers value rapid responses greatly. Users of mobile phones can get close-up access to information technology and cloud computing services thanks to mobile edge computing. In this article, the main goal is to use an artificial bee colony meta-innovative algorithm to solve the problem of placing edge servers in mobile edge computing. Moreover, load balancing between servers is one of the challenges discussed in this article. To deal with this issue, determination the locations of the servers using considering the distribution of workload between servers as a cost function in the artificial bee colony algorithm is a focused issue in this study. The results of the proposed method are compared with the load balancing criteria. The results of K-means compared to the clustering method show the superiority of the proposed method with regard to the loading criteria compared to this clustering method.**

Keywords*—Artificial bee colony algorithm, k-means, server placement, mobile cloud computing

I. INTRODUCTION

Now-a-days, smart mobile devices have become more critical for smart life [1-3]. In order to accelerate the growth of smart cities, it is necessary to increase the data transfer rate and the amount of infrastructure that is utilized by the services and application programs that make up the smart city [4, 5]. Due to the limited resources of mobile devices, mobile phone users do not experience the same levels of satisfaction as desktop device users, despite the fact that mobile applications are developing and becoming more computationally intensive [6]. Principally, we have a number of users in this issue. Users have a number of requests, and servers have the duty to respond to users' requests. Since each user has a specific position, each server must respond to the requests of users who are close to that server as much as possible. User requests are distributed between servers as a load. Loading part of the activities of mobile phone applications in remote-rich clouds is an effective technique to boost their performance [6–10]. However, because the cloud is frequently situated very distant from mobile phone users, there may be a significant and unpredictable delay in the data transfer between the two parties. It is undesirable for mobile applications, including augmented reality apps and mobile multiplayer gaming systems, where consumers value quick responses highly.

Mobile phone edge computing enables users of mobile phones to access cloud computing and information technology services inside radio access network coverage [11,12]. In order to decrease the delay, this technique moves the capacity for calculations and storage from the main network to the edge network. Edge servers can be installed nearby so that devices can offload part of the duties associated with their mobile applications, thus enhancing the quality of mobile user experiences.

Most studies have concentrated on offloading mobile user workloads to cloudlets to reduce mobile device energy consumption, and this strategy implies that clouds are already in place [13-16]. The impact of putting edge servers and loading mobile phone user workload onto them on mobile application performance has not received much consideration. In mobile edge computing, edge server deployment should take into account both the ideal location and the maximum number of edge servers. For this purpose, three points should be considered, which include the physical location of the edge servers, the planning of the number of edge servers, and the network deployment requirements. According to these three cases, in this article, we intend to choose the optimal location for deploying edge servers by using an artificial bee colony meta-initiative algorithm. The placement of the edge server in mobile edge computing environments has not received much research attention.

Numerous studies on cloudlet location have been conducted recently [17, 18]. The typical definition of a *cloud* is a network of computers that operate as a loader for users of mobile phones [19, 20]. Users of mobile phones can access edge servers close to base stations in the context of mobile edge computing [21]. To reduce the access delay between mobile phone users and distant clouds, edge servers can currently be considered download locations for mobile users. By sending calculations and storage capacity from the primary network to the Edge server, this problem is resolved [22]. It is believed that there are many similarities between cloudlet placement and edge servers [23, 24]. Mike Jia et al. [25] presented a load shedding system model for multi-user mobile task loading. They studied cloudlet placement and mobile user allocation to cloudlet. Then, with the aim of activating the location of the cloudlet in areas with high user density, they implemented an algorithm and, while balancing their workload, assigned mobile phone users to the contracted packages.

*Corresponding Author.

Zichuan Zhu and colleagues [16] utilized numerous wireless programs to investigate the cloudlet positioning issue. In order to reduce the access time between mobile users and cloudlets servicing users, they first defined this problem as a new cloudlet placement problem that assigns K locations to various key crucial sites. Second, they created an effective solution after defining the issue as integer linear programming. Studies [26, 27] about the computing burden for mobile edge cloud computing are also available. In a multi-channel wireless interference environment, Hu Chen et al. [28] investigated the issue of multi-user computing load for mobile edge cloud computing. They created a distributed computing offloading technique that attained the Nash equilibrium and characterized the distributed computing load decision problem for mobile phone users as a multi-user computing offloading game. Mike Jia et al. [25] only focused on the workload balance of cloudlets. Other studies [29-31] only on delay Access are examined, despite the fact that the aforementioned studies are successful in addressing the issue of cloudlet placement.

In this study, it is researched to find the position of servers that result in the optimization of coverage and balanced response to users' requests using the artificial bee colony meta-innovative method, which is motivated by the inspirations of previous studies. The steps of this study include studying the methods of the past, introducing the proposed method, and simulating this method. For this purpose, the MATLAB programming environment is used.

In this environment, the initial conditions for deploying edge servers in cloud computing are simulated using an artificial data set. In this simulation, variables such as the location of servers, base stations, and the number of user requests are considered. The results of this simulation are compared with the method based on K-means clustering. For the implementation of the Internet of Things, mobile edge computing has grown in importance [32]. Edge servers are smaller and less vulnerable to attacks since they are scattered over the mobile edge computing ecosystem. In addition, edge servers may function as private clouds to help with the issue of data leaking. There seems to be a need to employ a way to locate the server based on mobile edge computing technologies.

The rest of this paper presents the proposed method in Section II. The results and discussion discuss in Section III. The paper concludes in Section IV.

II. Proposed Method

In this section, a method for determining the location of servers in mobile edge computing is introduced. This method uses an artificial bee colony meta-initiative algorithm. In this method, by using this algorithm, we try to optimize a cost function, which leads to determining the location of the servers in such a way that the load balance between the servers is observed.

The primary objective of the problem of locating servers for mobile edge computing is to locate servers for proper and ideal user request response. The quantity of users must be taken into account first for this. These users each have a number of requests, which they submit to the server. The servers must respond to the requests of the users. In fact, each server receives a number of requests from users and must respond to these requests. According to the stated conditions, the location of the servers should be such that the servers follow the main goal. The main purpose of servers, in addition to responding to users' requests, is to distribute the load proportionally and optimally among themselves. In fact, servers should be located, so user requests are distributed among them. The position of the users and the number of requests of the users are known in this problem. However, the position of the servers should be answered as unknown in this problem. An artificial bee colony meta-innovative algorithm has been used to determine the location of the servers. The steps of the proposed method include:

defining and assigning values to the used variables,

creating a data set,

simulating the proposed method on the data set to determine the location of the servers, and

simulating the K-means clustering method on the data set to determine the location of the servers.

The comparison of the results obtained from the proposed method and the K-means clustering method with the load balancing criterion is introduced in the following stages of the proposed method.

A. Variables Under Study

In order to determine the status of the servers in the problem under consideration, the users and the requests of the users must be determined. According to the situation and the number of user requests, the values of the servers should be determined as unknown. In this study, the number of users, the position of users in the form of x and y coordinates in two-dimensional space, and the number of user requests are considered variables. Since real data is unavailable in this problem, synthetic data is used to value the studied variables. As an example, the position of the number of users is shown in Fig. 1.

The values of the users' coordinates are randomly assigned in the range [a-0]. The number of requests of each user is also a random number in the range of [1-w]. By assigning values to these variables, the used data set is produced.

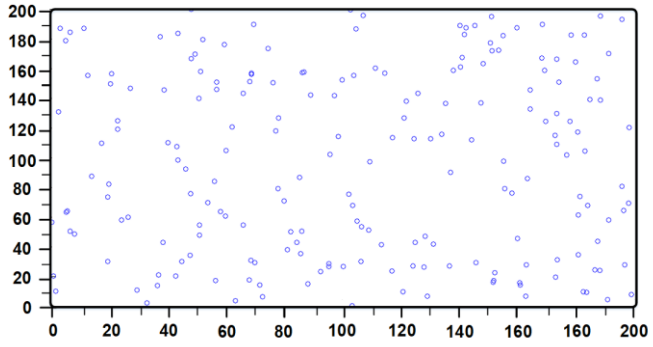

Fig. 1. The position of users in a specific area.

B. Artificial Bee Colony Algorithm to Servers Determine Location

In the issue of determining the location of the servers, it is necessary to determine the location of each server so that the servers can answer the requests of the users in a way that the division and distribution of the load are balanced between them. To determine load balance, it is necessary to have a relationship that determines the amount of load between servers, calculated according to Eq. (1). The artificial bee colony algorithm determines the optimal value for the load balance relationship [33].

$$WB = \sqrt{\frac{\sum_{i=1}^{k} (T^{i} - \bar{T})^2}{k}} \quad (1)$$

Where k represents the number of servers, T^i represents the number of requests that server i should respond. T indicates the average number of requests of all servers. In this regard, the lower the WB value indicates the existence of balance and load balance between servers. In order to respond to each user's request, a server must be assigned, and that server will answer the user's request. The location of servers and users are shown with two-dimensional coordinates. The closest server answers each user's request. Euclidean distance relationship is used to determine the distance between servers and a user. If we have k servers to respond to the request of user j according to equation (2), server i should be selected so that these servers have the smallest distance from user j.

$$\arg\min_{i \in (1,k)} \left(\sqrt{(x_j - x_i) + (y_j - y_i)}\right) \quad (2)$$

Where xi and yi represent the x and y position of server i, xj and yi represent the x and y position of server j. Since the goal and the unknown problem is to determine the location of the servers, by using the artificial bee colony algorithm, the initial location of the servers is generated as the initial population. By going through the steps of the bee colony algorithm to minimize equation (1) as a cost function, the most optimal position is determined for the servers. The stages of the artificial bee colony algorithm are according to Fig. 2.

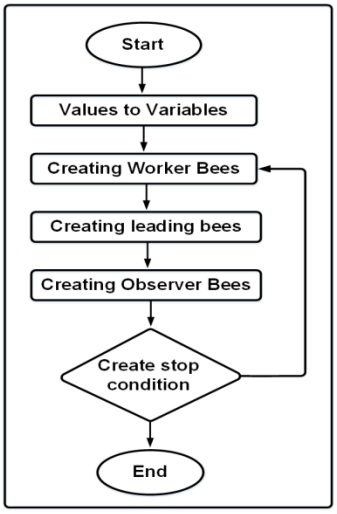

Fig. 2. Steps of artificial bee colony algorithm.

By having the location of server k, the closest server to that user is determined for each user using equation (2). After the closest server has been assigned to each user, the load balance between the servers can be determined by equation (2). For each population produced in the steps of the artificial bee colony algorithm, the fitness of this population is also determined by equation (2). The stages of production of worker bees, observers, and leaders lead to searching in the state space and producing better answers to the investigated problem.

C. Proposed Method Evaluation

In the problem of determining the location of the servers in calculating the mobile edge, two important criteria and metrics for the location of the servers are considered. The combination of these two criteria, which includes the determination of the closest server to each user and the distribution of user requests, leads to the creation of a standard called load balance. Since the servers have to perform the users' tasks in the end, the servers' location should be such that all servers can be balanced in terms of workload. In the problem of determining the location of servers, load balancing means balanced distribution of user requests between servers. The proposed method uses the load balance criterion as a cost function in the artificial bee colony algorithm. The results obtained from the proposed method have been compared with the K-means clustering method to evaluate the proposed method. In the K-means clustering method, users are placed in k clusters, and each cluster's center is considered the server's location. In fact, the K-means clustering algorithm groups two-dimensional points that are the coordinates of users into k clusters. Users in a cluster send their requests to the server head of the current cluster. With this assumption, the load balance is also determined by the K-means clustering method according to equation (1). The evaluation of the proposed and K-means clustering methods is done by comparing the load balance value. The method with less load balance is the more suitable method for determining the location of the servers.

III. Result and Discussion

In this section, the simulation of the proposed method is examined. According to the proposed method stated in the previous section, the steps of the proposed method are simulated in this section. This section includes the introduction of the data set, performing some tests on the data set, and reviewing the results obtained from the various tests.

A. Dataset Used

To simulate the proposed method, it is necessary to set and select the variables and data set. The first part of the data set is the simulation environment. In the simulation environment, the position of users and servers is determined. The simulation environment is a square with side a. The second part of the data set is the users. Variables related to users include the number of users, the position of each user, and the number of requests for each user. The third part of the data set is the servers. The variables related to servers also include the number of servers and the location of each server. The required variables are introduced in Table I.

The position of each user is shown as a point in two-dimensional coordinates. The value of x for the i-th user is in a

specific range in the interval between $xf-xe$. The value of y for the i-th user is also within the specified range in the interval $yf-ye$. Each server also has a location. The position of each server, like the position of each user, is shown as a point in two-dimensional space. The range of x and y values for each server is the same as the range of these values for users. Each user also has a number of requests. The number of requests of user i is determined by a number in the range $wf-we$. Since users and servers are shown with two-dimensional coordinates, the distance between a user *i* and server *j* is represented by the symbol dij. Euclidean distance, according to Eq. (1) is used to calculate the distance between a user and a server.

The position of each user and the number of requests of each user are randomly determined within the specified range. The position of each server is also unknown and is calculated and determined by the proposed method. In the proposed method, an artificial bee colony meta-initiative algorithm is used. The variables of this algorithm are the number of repetitions and the number of the initial population. The values of these variables are shown in Table II.

B. Results Evaluation

In this section, the variables stated in the previous section are set. Having the data set in this section, several different tests are performed. In these experiments, the proposed K-means clustering methods are compared with each other. These different experiments which are simulating and determining the position of the servers. The main difference is in the valuing of the studied variables. This section consists of the results of three experiments. In these tests, the number of users, the position of users, and the range determined for the position of users are considered different from each other. The values of the variables in the first experiment are according to Table III.

The results obtained from this experiment with four different executions are according to Table IV. This experiment has been done with four different implementations.

The position of users and servers in each of these four tests in the proposed method and the K-means clustering method is according to Fig. 3 to 10.

The values of the cost function in the artificial bee colony algorithm are according to Fig. 11 to 14.

In the first experiment with the criterion of workload balance, the results of the proposed method are better than the k-means clustering method. In this test, even in Fig. 7, the selection of two servers with a close location shows that the proposed method has considered even close locations in the search space. Although this happens randomly, the ability to search in the state space in the proposed method is better than the k-means method. In these four implementations, the proximity of the servers to each other or the distance between the servers did not have such a big effect on the results of the load balance. In other words, the artificial bee colony method in different repetitions is able to find several answers in different situations, but with the results of the load balance approximately to find the same. In contrast, in the k-means method, the distance between the servers in different executions is not so different from each other. Although the answers generated for the location of the servers are different, the distance between the servers in each of these four executions is greater than the executions of the proposed method. In the graphs of the cost function values in the proposed method, the proposed method has converged in many iterations. The values of the variables in the second experiment are according to Table V.

The results obtained from this experiment with four different executions are according to Table VI.

Each of these four tests in the proposed method and the k-means clustering method is according to Fig. 15 to 26.

In the second experiment, with the criterion of workload balance, the results of the proposed method are better than the k-means clustering method. In this test, the number of users and requests are the same as in the first test, but the position of the users is randomly assigned in a larger range than in the first test. The results of the load balance values in this experiment are almost similar to the values in the first experiment since the number of user requests did not differ from the first experiment. In this case, it is only proportional to the situation of new users of the servers. The proposed method and the k-means method determine that the servers are positioned so that the load balance values are not much different from the first test. The position of the servers in the proposed method is better than the k-means method in terms of distance variation in each execution. In fact, the proposed method has produced answers that the results of the location of the servers are close to each other in some executions and far from each other in others. Despite the production of various answers, the results of load balance in different implementations with the proposed method are not much different. The values of the variables in the third experiment are according to Table VII.

The results obtained from this experiment with four different executions are according to Table VIII.

Each of these four tests in the proposed method and the k-means clustering method is according to Fig. 27 to 34.

The values of the cost function in the artificial bee colony algorithm are according to Fig. 35 to 38.

In the third experiment with the criterion of workload balance, the results of the proposed method are better than the k-means clustering method. In this experiment, the number of users, the position of the users, and the number of user requests differ from the first experiment. The number of users in this test is twice the number in the first test. The position that can be selected for assigning values to the coordinates of users is also twice the first test. The number of requests of 200 users in the third test is similar to the first. That is, the number of requests of the first 100 users in the third test is similar to the 100 users of the first test. The number of requests of the second 100 users is similar to the 100 users of the first test. In this experiment, the location of the servers in the proposed method is better than the k-means method in terms of distance variation in each execution. In fact, the proposed method has produced answers that the results of the location of the servers are close to each other in some executions and far from each other in others. Despite the production of various answers, the results of load balance in different executions with the proposed method are not much different. In this section, the main goal is to

simulate the proposed method. The results of the proposed method have been compared with the k-means clustering method with the criterion of load balance. In order to simulate the proposed method, three tests have been performed in this section. The results obtained from these tests show the superiority of the proposed method over the k-means clustering method with load balancing criteria and generating various answers for the servers' location.

TABLE I. VARIABLES USED

Variable Name	Variable Symbol
Limited simulation environment	A square with side a
Number of users	n
The position of each user i	$x\,i \in [x\,f - x\,e]$ $y\,i \in [y\,f - y\,e]$
Number of requests per user i	$w\,i \in [y\,f - y\,e]$
Number of servers	k
Server position j	$x\,i \in [x\,f - x\,e]$ $y\,i \in [x\,f - x\,e]$
The distance between user i and server j	$d\,i\,j$

TABLE II. THE VALUES OF THE VARIABLES OF AN ARTIFICIAL BEE COLONY META-INITIATIVE ALGORITHM

Artificial bee colony algorithm variable name	Value
Number of repetitions	50
Primary population	10

TABLE III. THE VALUES OF THE VARIABLES IN THE FIRST EXPERIMENT

Variable Name	Variable Symbol
Limited simulation environment	A square with side 200
Number of users	100
The position of each user i	$x\,i \in [0-200]$ $y\,i \in [0-200]$
Number of requests per user i	$w\,i \in [1-5]$
Number of servers	5

TABLE IV. THE RESULTS OF THE FIRST EXPERIMENT

Tests Name	WB_{acb}	WB_{kmeans}
The first test, the first performance	3.6	13.3026
The first test of the second performance	4.0694	19.3742
The first test of the third performance	4.6648	8.8408
The first test of the fourth performance	3.763	9.0863

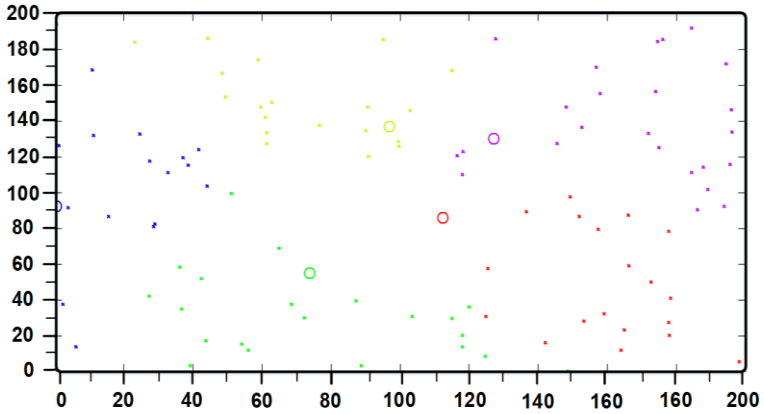

Fig. 3. The position of the servers in the first test of the first execution with the proposed method.

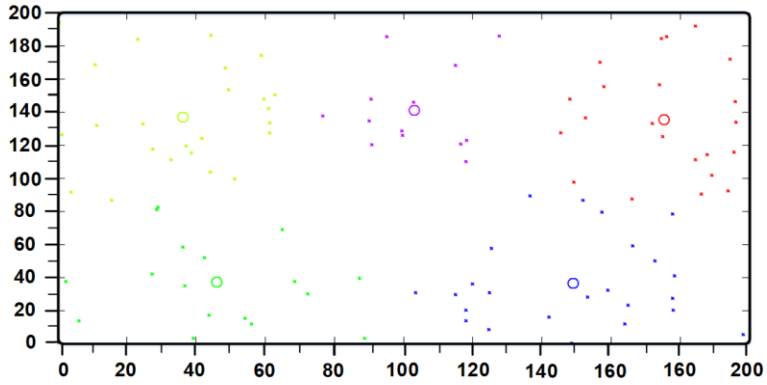

Fig. 4. The position of the servers in the first test of the first run with the k-means method.

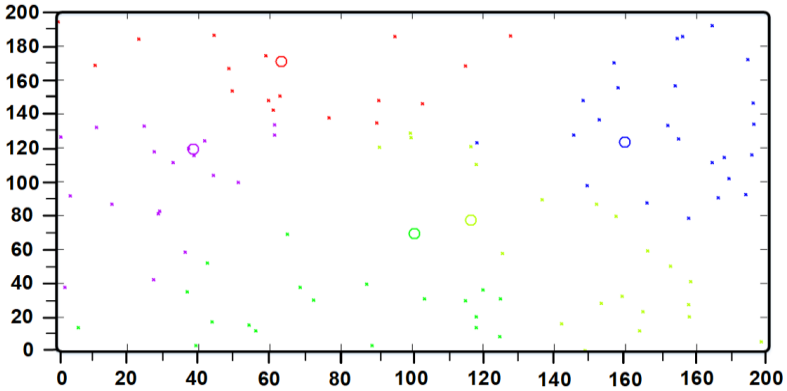

Fig. 5. The position of the servers in the first test of the second execution with the proposed method.

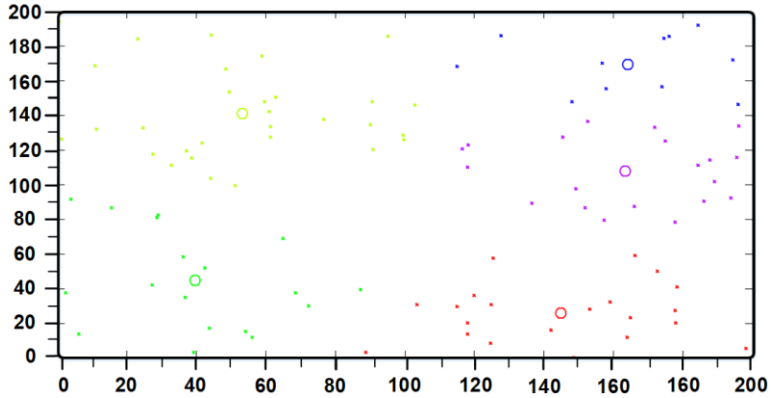

Fig. 6. The position of the servers in the first test of the second run with the k-means method.

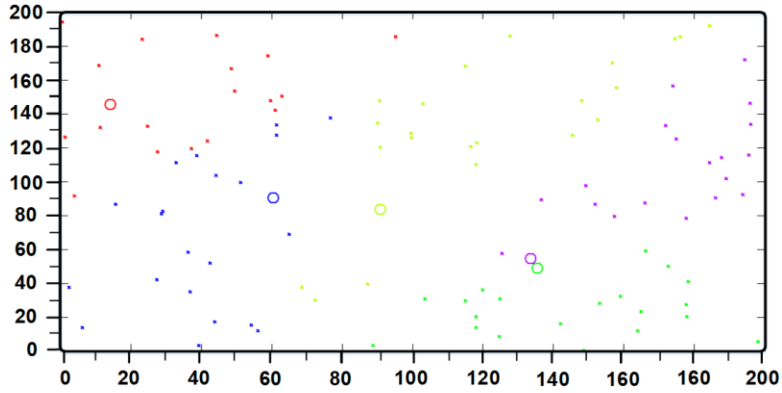

Fig. 7. The position of the servers in the first test of the third execution with the proposed method.

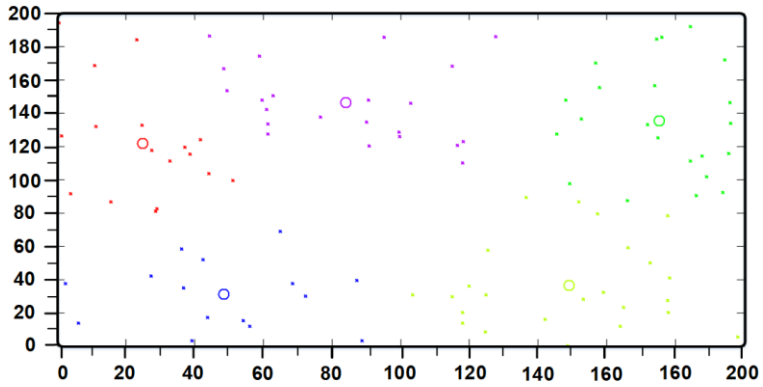

Fig. 8. The position of the servers in the first test of the third run with the k-means method.

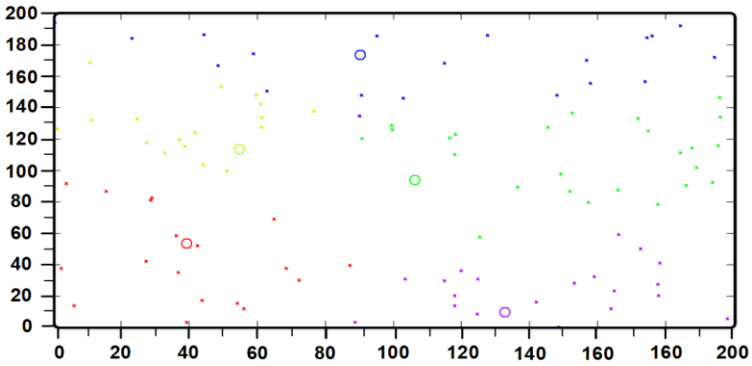

Fig. 9. The position of the servers in the first test of the fourth implementation with the proposed method.

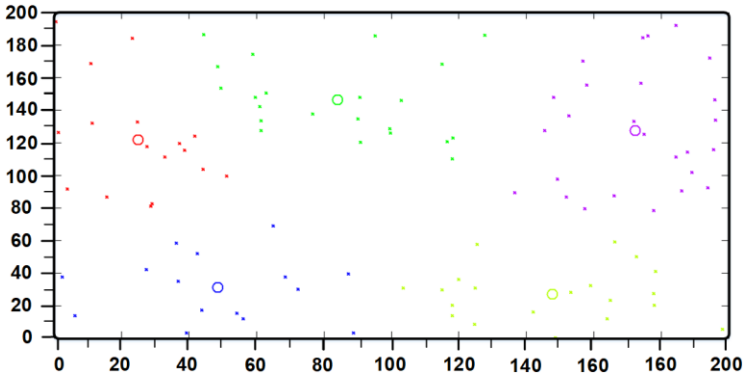

Fig. 10. The position of the servers in the first test of the fourth run with the k-means method.

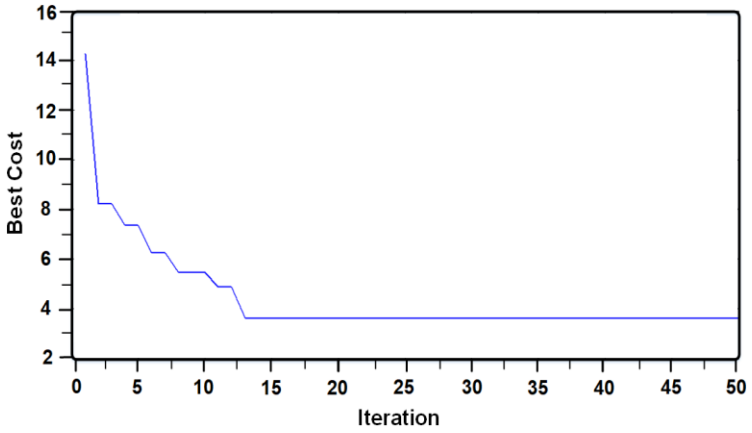

Fig. 11. The value of the cost function in the first experiment of the first execution with the proposed method.

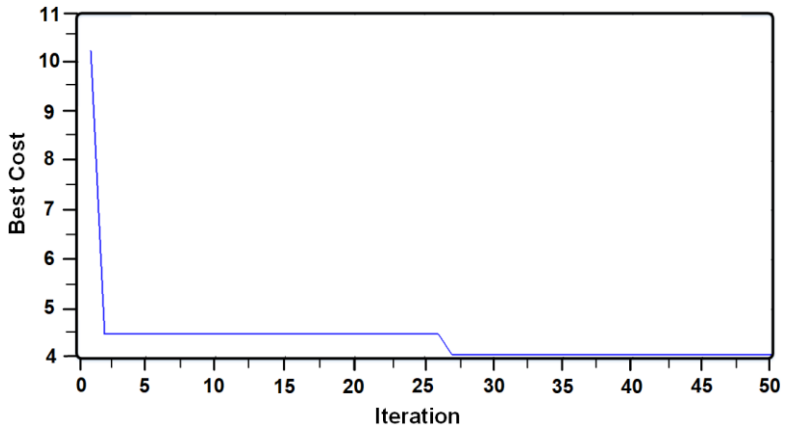

Fig. 12. The value of the cost function in the first experiment of the second implementation with the proposed method.

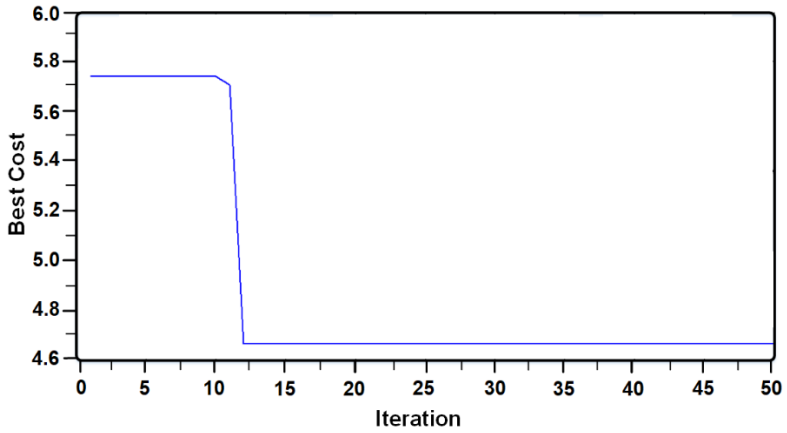

Fig. 13. The value of the cost function in the first experiment of the third implementation with the proposed method.

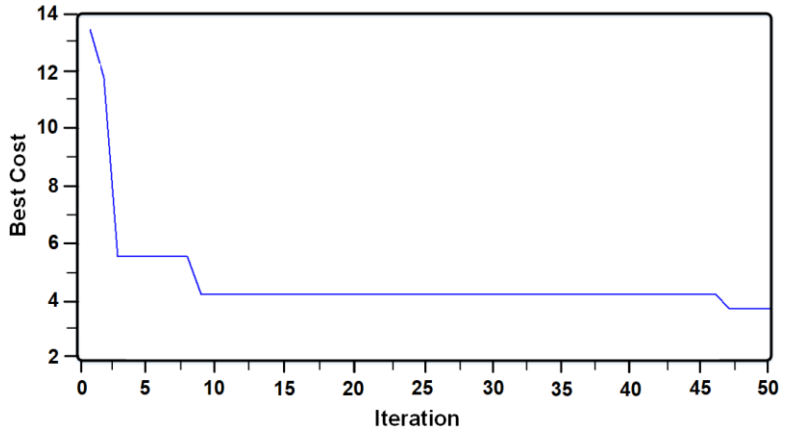

Fig. 14. The value of the cost function in the first experiment of the fourth implementation with the proposed method.

TABLE V. THE VALUES OF THE VARIABLES IN THE SECOND EXPERIMENT

Variable Name	Variable Symbol
Limited simulation environment	A square with side 500
Number of users	100
The position of each user i	$x\,i \in [0-500]$ $y\,i \in [0-500]$
Number of requests per user i	$w\,i \in [1-5]$
Number of servers	5

TABLE VI. THE RESULTS OF THE SECOND EXPERIMENT

Tests Name	WB_{acb}	WB_{kmeans}
The second test, the first performance	3.9699	10.4957
The second test of the second performance	4.2615	10.4957
The second test of the third performance	3.0594	11.6516
The second test of the third performance	4.9558	10.4957

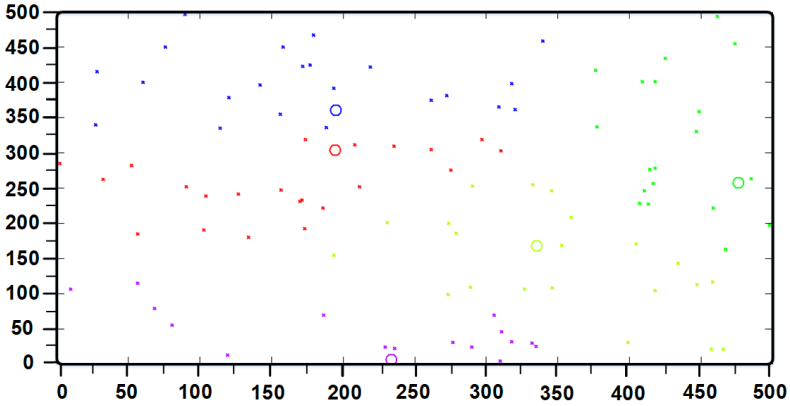

Fig. 15. The position of the servers in the second test of the first execution with the proposed method.

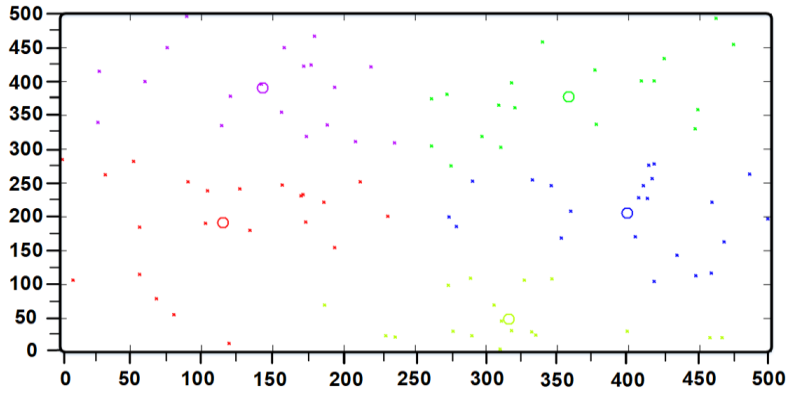

Fig. 16. The position of the servers in the second test of the first run with the k-means method.

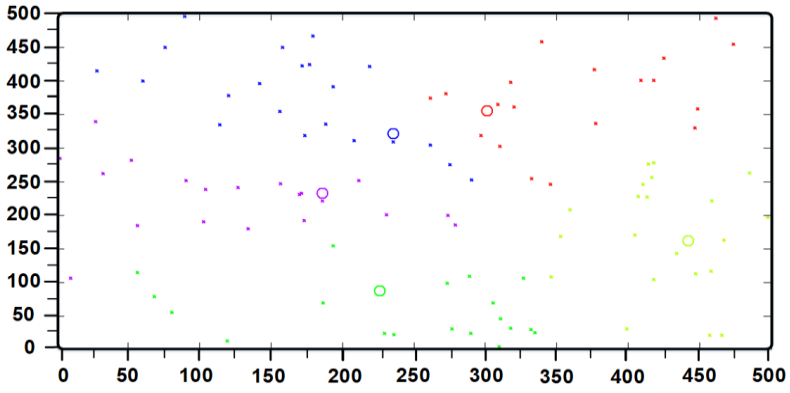

Fig. 17. The position of the servers in the second test of the second execution with the proposed method.

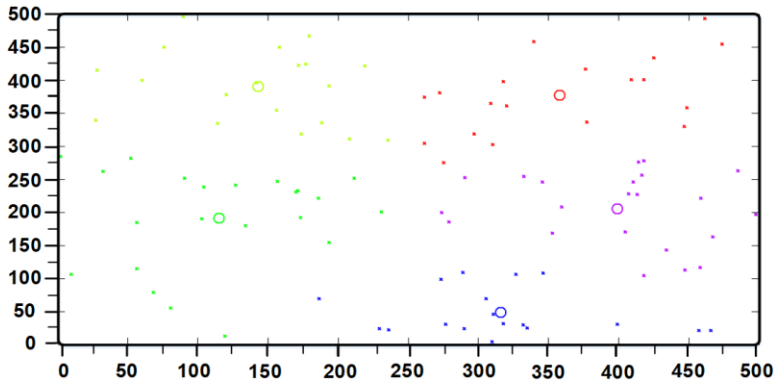

Fig. 18. The position of the servers in the second test of the second run with the k-means method.

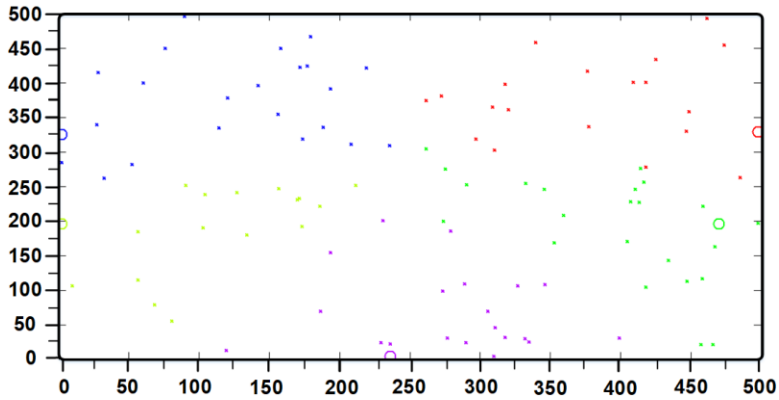

Fig. 19. The position of the servers in the second test of the third execution with the proposed method.

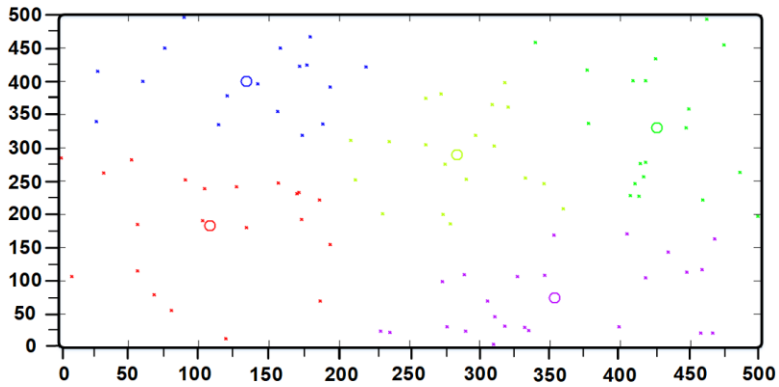

Fig. 20. The position of the servers in the second test of the third run with the k-means method.

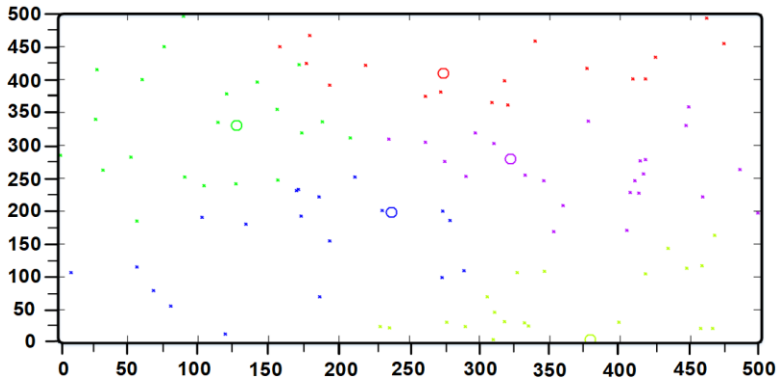

Fig. 21. The position of the servers in the second test of the fourth implementation with the proposed method.

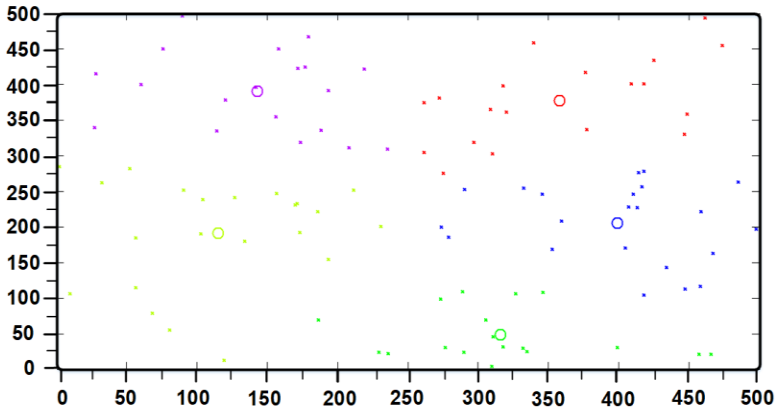

Fig. 22. The position of the servers in the second test of the fourth run with the k-means method.

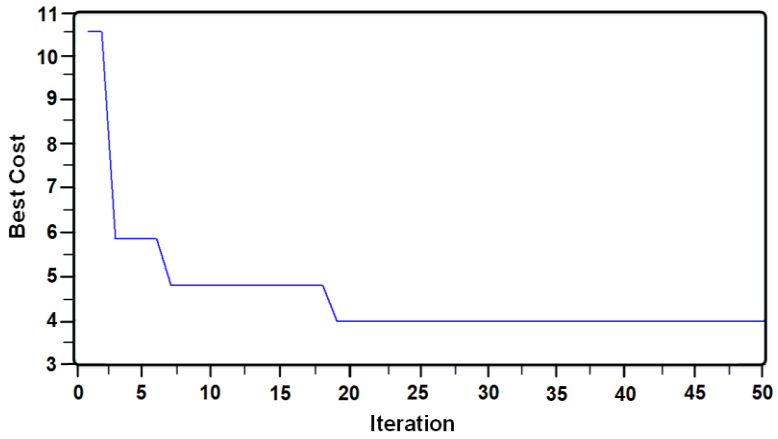

Fig. 23. The value of the cost function in the second experiment of the first execution with the proposed method.

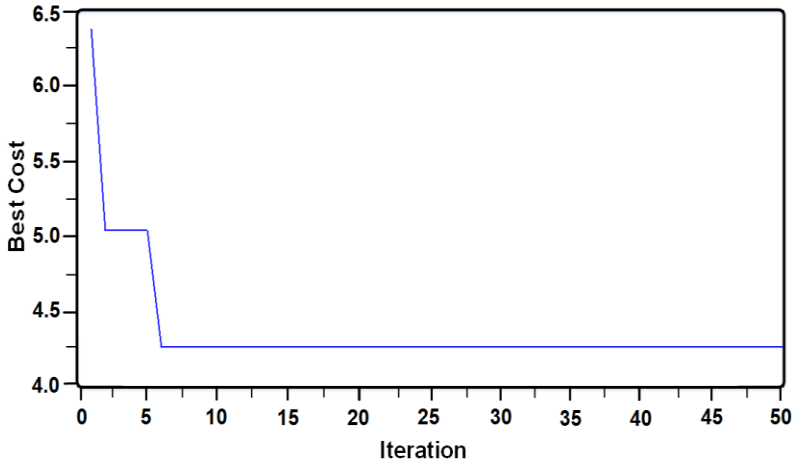

Fig. 24. The value of the cost function in the second experiment of the second implementation with the proposed method.

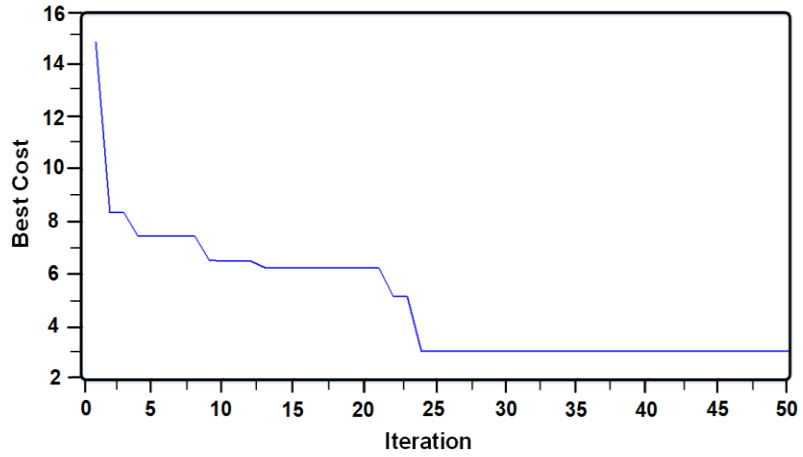

Fig. 25. The value of the cost function in the second experiment of the third implementation with the proposed method.

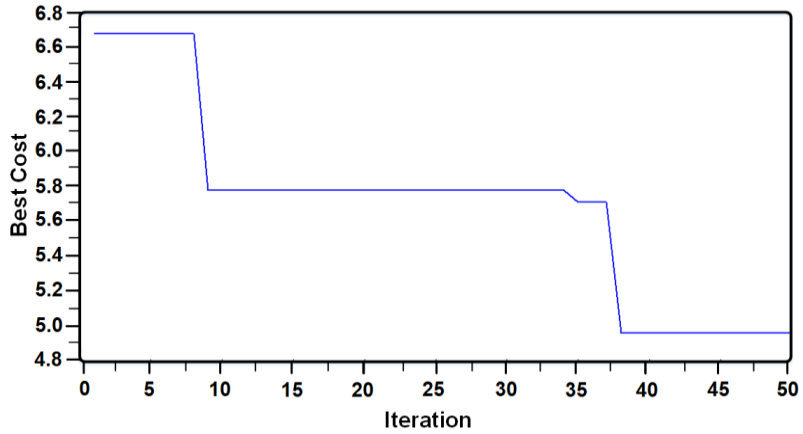

Fig. 26. The value of the cost function in the second experiment of the fourth implementation with the proposed method

TABLE VII. THE VALUES OF THE VARIABLES IN THE THIRD EXPERIMENT

Variable Name	Variable Symbol
Limited simulation environment	A square with side 200
Number of users	200
The position of each user i	$x\,i \in [0-200]$ $y\,i \in [0-200]$
Number of requests per user i	The first 100 users are the same as the first 100 test users, the second 100 users are the same as the first 100 test users
Number of servers	5

TABLE VIII. THE RESULTS OF THE THIRD EXPERIMENT

Tests Name	WB_{acb}	WB_{kmeans}
The third test, the first performance	6.7705	25.9892
The third test of the second performance	10.1902	25.6562
The third test of the third performance	5.2383	22.6945
The third test of the third performance	8.1633	22.2764

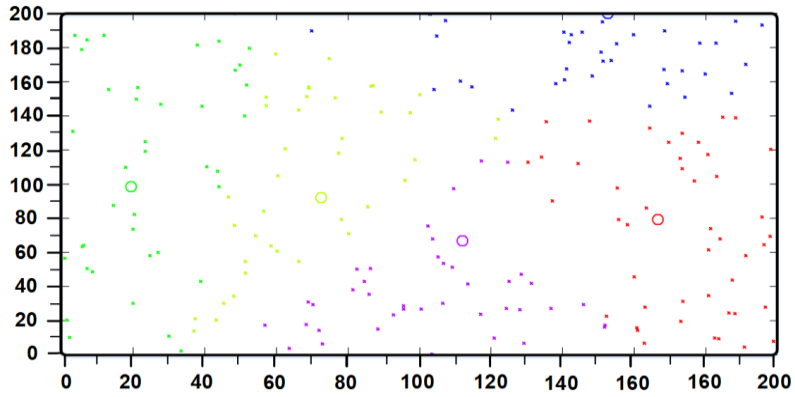

Fig. 27. The position of the servers in the third test of the first execution with the proposed method.

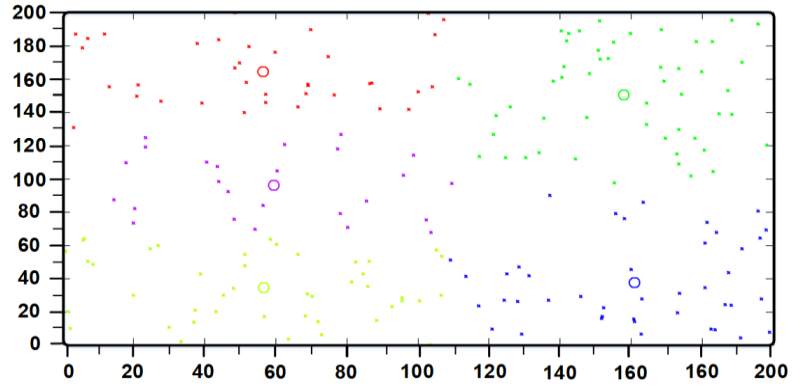

Fig. 28. The position of the servers in the third test of the first run with the k-means method.

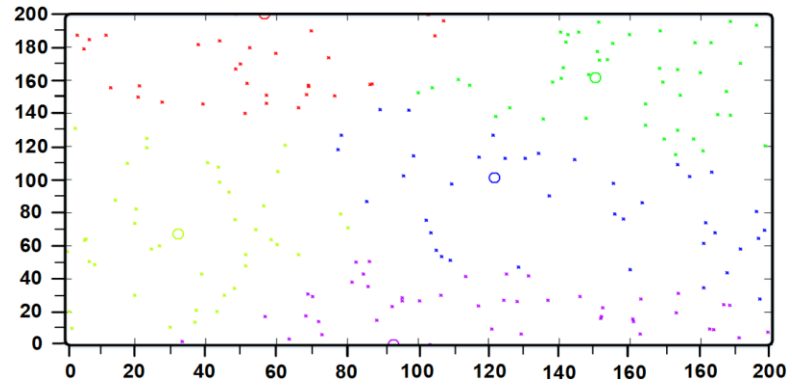

Fig. 29. The position of the servers in the third test of the second execution with the proposed method.

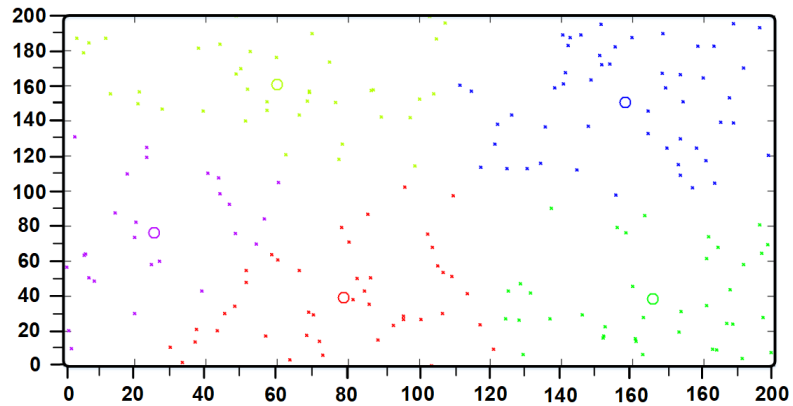

Fig. 30. The position of the servers in the third test of the second run with the k-means method.

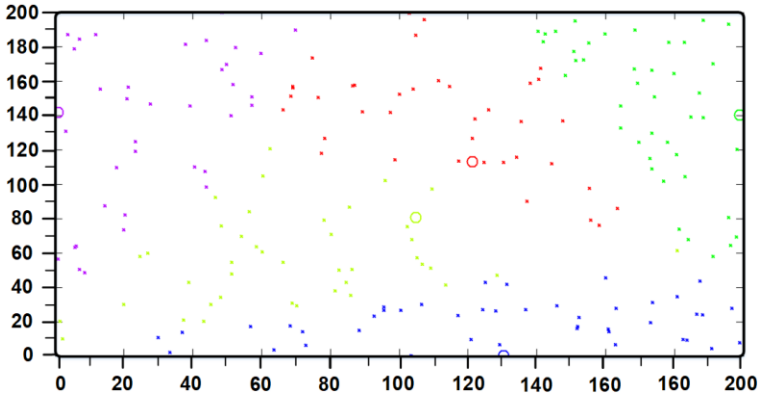

Fig. 31. The position of the servers in the third test of the third execution with the proposed method.

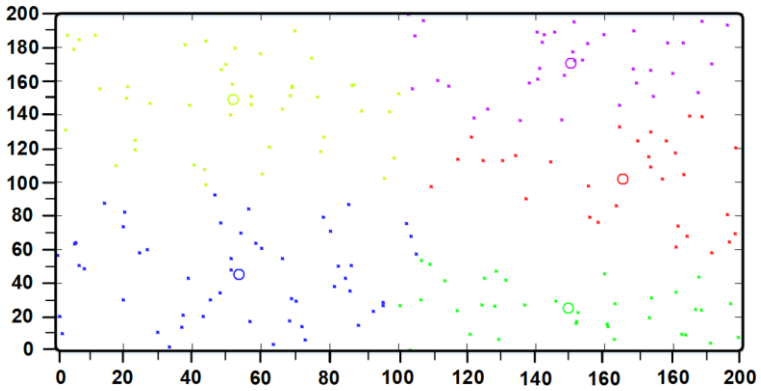

Fig. 32. The position of the servers in the third test of the third run with the k-means method.

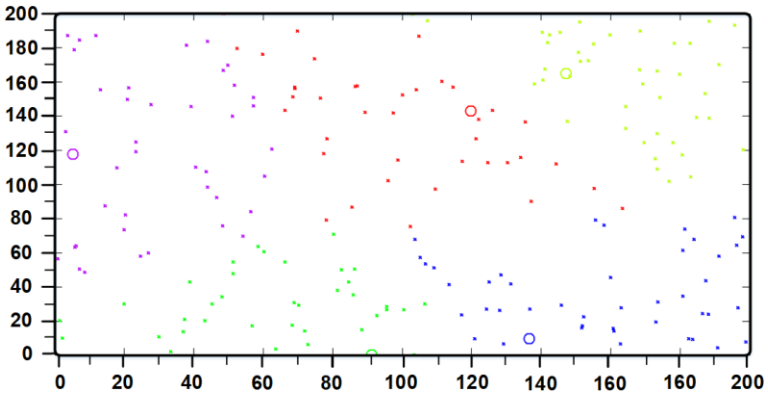

Fig. 33. The position of the servers in the third test of the fourth implementation with the proposed method.

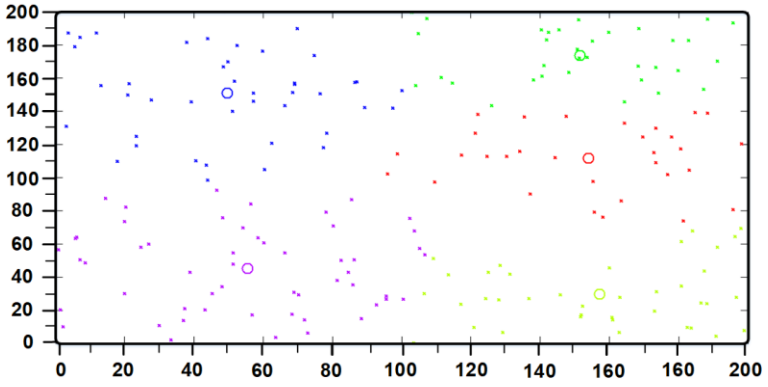

Fig. 34. The position of the servers in the third test of the fourth run with the k-means method.

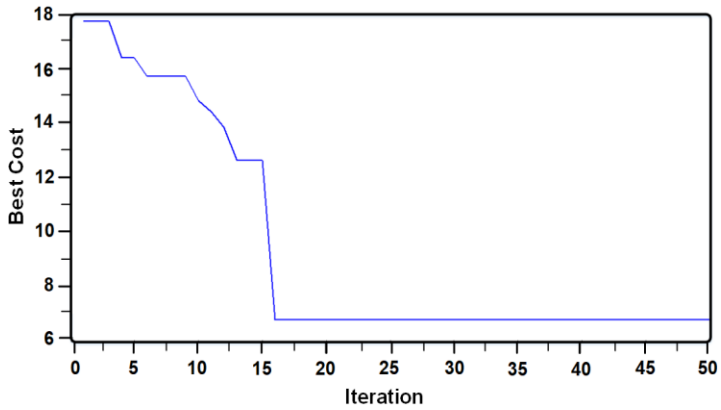

Fig. 35. The value of the cost function in the third experiment of the first execution with the proposed method.

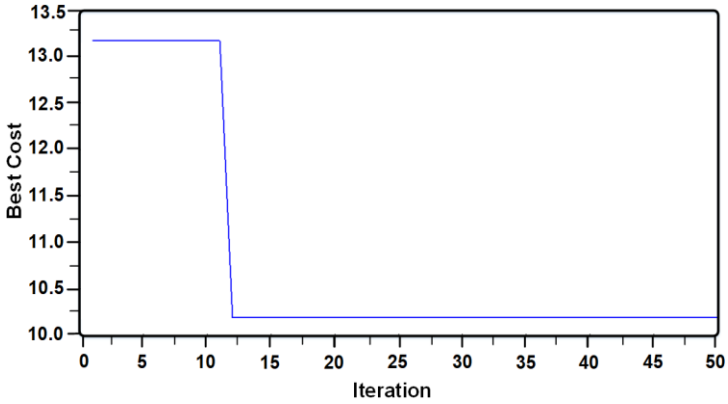

Fig. 36. The value of the cost function in the third experiment of the second implementation with the proposed method.

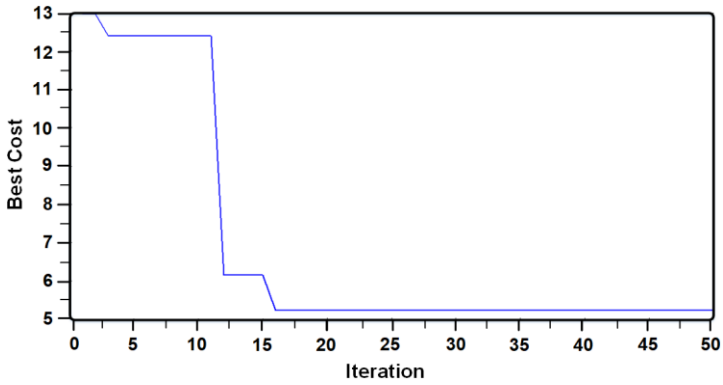

Fig. 37. The value of the cost function in the third experiment of the third implementation with the proposed method.

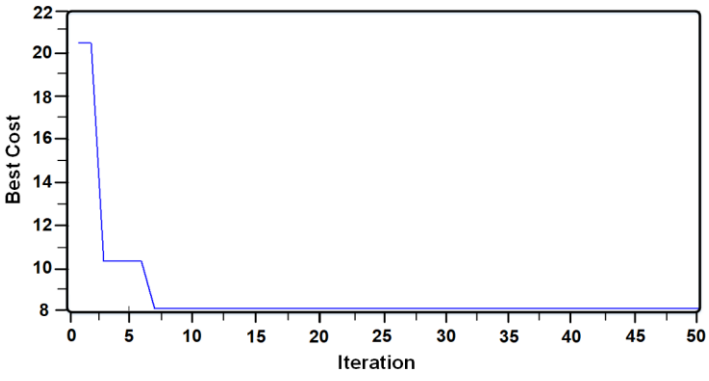

Fig. 38. The value of the cost function in the third experiment of the fourth implementation with the proposed method.

IV. CONCLUSION

In the problem of determining the location of the servers, it should be possible to determine the location of the servers in the edge computing environment in mobile computing in such a way that by having k servers, these servers respond to the requests of the users in such a way that the load of the servers is balanced. In other words, the servers should be in a position to be able to respond to the requests of nearby users, and the number of requests that the servers respond to should be balanced. A criterion called load is used to balance and distribute user requests between servers. The major contribution of this study is to employ a meta-innovative artificial bee colony algorithm to address the issue of where to locate edge servers for mobile edge computing. Moreover, one of the difficulties covered in this essay is load balancing between servers. This study's main focus is on determining the server placements using the artificial bee colony method while taking workload distribution between servers into account as a cost function. The proposed method's outcomes are contrasted with the load balancing criterion. The comparison of K-means results to the clustering approach demonstrates the proposed method presented results superiority with regard to the loading criteria. For future study, the proposed method can be implemented in real scenarios. Furthermore, other optimization algorithms including Particle Swarm Optimization (PSO) can be investigated and the result compared to current study to address the better solution.

ACKNOWLEDGMENTS

Training plan for young backbone teachers in Henan Province (No.2018GGJS267).

REFERENCES

[1] Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18), PP 1587-1611.

[2] Taleb, T., Dutta, S., Ksentini, A., Iqbal, M., & Flinck, H. (2017). Mobile edge computing potential in making cities smarter. IEEE Communications Magazine, 55(3), PP 38-43.

[3] Sabella, D., Vaillant, A., Kuure, P., Rauschenbach, U., & Giust, F. (2016). Mobile-edge computing architecture: The role of MEC in the Internet of Things. IEEE Consumer Electronics Magazine, 5(4), PP 84-91.

[4] Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012, March). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In 2012 Proceedings IEEE Infocom (pp. 945-953). IEEE.

[5] Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2010, October). Cuckoo: a computation offloading framework for smartphones. In International Conference on Mobile Computing, Applications, and Services (pp. 59-79). Springer, Berlin, Heidelberg.

[6] Chun, B. G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011, April). Clonecloud: elastic execution between mobile device and cloud. In Proceedings of the sixth conference on Computer systems (pp. 301-314).

[7] Ahmed, E., Gani, A., Sookhak, M., Ab Hamid, S. H., & Xia, F. (2015). Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. Journal of Network and Computer Applications, 52, PP 52-68.

[8] Patel, M., Naughton, B., Chan, C., Sprecher, N., Abeta, S., & Neal, A. (2014). Mobile-edge computing introductory technical white paper. White paper, mobile-edge computing (MEC) industry initiative, 29, PP 854-864.

[9] Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010, June). Maui: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 49-62).

[10] Beck, M. T., Feld, S., Fichtner, A., Linnhoff-Popien, C., & Schimper, T. (2015, February). ME-VoLTE: Network functions for energy-efficient video transcoding at the mobile edge. In 2015 18th International Conference on Intelligence in Next Generation Networks (pp. 38-44). IEEE.

[11] Taleb, T., Dutta, S., Ksentini, A., Iqbal, M., & Flinck, H. (2017). Mobile edge computing potential in making cities smarter. IEEE Communications Magazine, 55(3), PP 38-43.

[12] Li, H., Dong, M., Ota, K., & Guo, M. (2016). Pricing and repurchasing for big data processing in multi-clouds. IEEE Transactions on Emerging Topics in Computing, 4(2), PP 266-277.

[13] Ahmed, E., Akhunzada, A., Whaiduzzaman, M., Gani, A., Ab Hamid, S. H., & Buyya, R. (2015). Network-centric performance analysis of runtime application migration in mobile cloud computing. Simulation Modelling Practice and Theory, 50, PP 42-56.

[14] Chun, B. G., & Maniatis, P. (2010, June). Dynamically partitioning applications between weak devices and clouds. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (pp. 1-5).

[15] Xu, Z., Liang, W., Xu, W., Jia, M., & Guo, S. (2015). Efficient algorithms for capacitated cloudlet placements. IEEE Transactions on Parallel and Distributed Systems, 27(10), PP 2866-2880.

[16] Jia, M., Cao, J., & Liang, W. (2015). Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Transactions on Cloud Computing, 5(4), PP 725-737.

[17] Magurawalage, C. M. S., Yang, K., Hu, L., & Zhang, J. (2014). Energy-efficient and network-aware offloading algorithm for mobile cloud computing. Computer Networks, 74, PP 22-33.

[18] Yao, H., Bai, C., Xiong, M., Zeng, D., & Fu, Z. (2017). Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. Concurrency and Computation: Practice and Experience, 29(16), e3975.

[19] Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C., & Zhang, J. C. (2014). What will 5G be?. IEEE Journal on selected areas in communications, 32(6), PP 1065-1082.

[20] Morgado, A., Huq, K. M. S., Mumtaz, S., & Rodriguez, J. (2018). A survey of 5G technologies: regulatory, standardization and industrial perspectives. Digital Communications and Networks, 4(2), PP 87-97.

[21] Wang, T., Zhou, J., Zhang, G., Wei, T., & Hu, S. (2019). Customer perceived value-and risk-aware multiserver configuration for profit maximization. IEEE Transactions on Parallel and Distributed Systems, 31(5), PP 1074-1088.

[22] Liu, Y., Peng, M., Shou, G., Chen, Y., & Chen, S. (2020). Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things. IEEE Internet of Things Journal, 7(8), PP 6722-6747.

[23] Chen, M., Guo, S., Liu, K., Liao, X., & Xiao, B. (2020). Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. IEEE Transactions on Mobile Computing, 20(5), PP 2025-2040.

[24] Chen, M., Guo, S., Liu, K., Liao, X., & Xiao, B. (2020). Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. IEEE Transactions on Mobile Computing, 20(5), PP 2025-2040.

[25] Xiang, H., Xu, X., Zheng, H., Li, S., Wu, T., Dou, W., & Yu, S. (2016, December). An adaptive cloudlet placement method for mobile applications over GPS big data. In 2016 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[26] Clinch, S., Harkes, J., Friday, A., Davies, N., & Satyanarayanan, M. (2012, March). How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users. In 2012 IEEE international conference on pervasive computing and communications (pp. 122-127). IEEE.

[27] Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., & Qureshi, A. (2015). Application partitioning algorithms in mobile cloud computing:

Taxonomy, review and future directions. Journal of Network and Computer Applications, 48, PP 99-117.

[28] Gu, F., Niu, J., Qi, Z., & Atiquzzaman, M. (2018). Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions. Journal of Network and Computer Applications, 119, PP 83-96.

[29] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4), PP 14-23.

[30] Wolbach, A., Harkes, J., Chellappa, S., & Satyanarayanan, M. (2008, June). Transient customization of mobile computing infrastructure. In Proceedings of the First Workshop on Virtualization in Mobile Computing (pp. 37-41).

[31] Tao, M., Ota, K., & Dong, M. (2017). Foud: Integrating fog and cloud for 5G-enabled V2G networks. IEEE Network, 31(2), PP 8-13.

[32] Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012, March). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In 2012 Proceedings IEEE Infocom (pp. 945-953). IEEE.

[33] Wang, S., Zhao, Y., Xu, J., Yuan, J., & Hsu, C. H. (2019). Edge server placement in mobile edge computing. Journal of Parallel and Distributed Computing, 127, PP 160-168.