Meta-heuristic-based offloading task optimization in mobile edge computing

Aamir Abbas¹, Ali Raza², Farhan Aadil¹ and Muazzam Maqsood¹

Abstract
With the recent advancements in communication technologies, the realization of computation-intensive applications like virtual/augmented reality, face recognition, and real-time video processing becomes possible at mobile devices. These applications require intensive computations for real-time decision-making and better user experience. However, mobile devices and Internet of things have limited energy and computational power. Executing such computationally intensive tasks on edge devices either leads to high computation latency or high energy consumption. Recently, mobile edge computing has been evolved and used for offloading these complex tasks. In mobile edge computing, Internet of things devices send their tasks to edge servers, which in turn perform fast computation. However, many Internet of things devices and edge server put an upper limit on concurrent task execution. Moreover, executing a very small size task (1 KB) over an edge server causes increased energy consumption due to communication. Therefore, it is required to have an optimal selection for tasks offloading such that the response time and energy consumption will become minimum. In this article, we proposed an optimal selection of offloading tasks using well-known metaheuristics, ant colony optimization algorithm, whale optimization algorithm, and Grey wolf optimization algorithm using variant design of these algorithms according to our problem through mathematical modeling. Executing multiple tasks at the server tends to provide high response time that leads to overloading and put additional latency at task computation. We also graphically represent the tradeoff between energy and delay that, how both parameters are inversely proportional to each other, using values from simulation. Results show that Grey wolf optimization outperforms the others in terms of optimizing energy consumption and execution latency while selected optimal set of offloading tasks.

Keywords
Mobile edge computing, MCC, energy optimization in mobile edge computing, offloading decision in mobile edge computing

Date received: 30 November 2020; accepted: 12 May 2021

Handling Editor: Dr Yanjiao Chen

Introduction
Internet of things (IoT) is a pretty research vibrant area since the last decade. With the evolution of IoT, mobile users (including smartphones and laptops) have set off a new wave. It is considered as driving power for the success and betterment of these smart devices.¹ The realization of computationally intensive applications also becomes possible at mobile devices. IoT has been successfully used in the heterogeneous set of applications

¹Computer Science Department, COMSATS University Islamabad, Islamabad, Pakistan
²Department of Computer Science, University of Engineering and Technology, Taxila, Taxila, Pakistan

Corresponding author:
Farhan Aadil, Computer Science Department, COMSATS University Islamabad, Attock Campus, Islamabad 43600, Pakistan.
Email: farhan.aadil@ciit-attock.edu.pk
such as autonomous driving, intelligent transportation system, virtual/augmented reality, face recognition, and video processing systems. Most of these applications require heavy computations for real-time decision-making and better user experience. Moreover, these applications have a diverse set of requirements. Some of these require ultra-low latency, whereas other demand for optimal energy consumption to prolong their operational life. However, mobile and IoT devices have imperfect battery energy and limited computational power. Executing such heavy computationally intensive tasks on these devices either extends the processing delay or produces the results in each time bound at the cost of higher energy consumption. Figure 1 illustrates the case scenario of the mobile cloud computing (MCC) model for the image recognition system. The mobile device captures the image and sends data to the cloud server. The server performs face detection, pre-processing, and classification of the image, and reports back to the mobile device for further processes.

Edge servers have limited computational resources. Although the edge server’s computational power is much greater than the mobile devices, it is relatively too low as compared to the cloud server. Therefore, the large number of IoT and limited computational power of edge server put an upper bound on concurrent task execution. Executing a huge number of tasks at the server tends to provide high response time. If all the nodes send their data for offload computation, the edge server will become overloaded and put additional latency at task computation. Moreover, if the task size is too small, executing it over an edge server causes increased energy consumption due to communication. For delay-sensitive applications where IoT has limited battery energy, offloading a task to the server is only viable if the following conditions are met:

- When the total time cost to compute task at mobile device is greater than the total time to compute the task at mobile edge computing (MEC) server.
- When the total energy consumption at mobile device is greater than the energy consumed for offloading the task.

To fully exploit the benefits of MEC, it is required to have an optimal selection for offloading tasks. This optimal selection leads to minimize the response time and energy consumption of IoT devices.

In this article, we use well-known evolutionary algorithms, that is, ant colony optimization (ACO), Grey wolf optimization (GWO), and whale optimization algorithm (WOA), to get an optimal selection of offloading task. To perform the selective offloading with multiple input tasks such that the overall network energy consumption of IoT and the overall network delay become minimum. The reason to use evolutionary algorithms in this optimization problem is as follows:

- Evolutionary algorithms are the best for solving multi-objective tasks. Energy and delay minimization along with the identification of the offloading tasks are the multi-objectives to achieve.
- It is also useful to check whether our algorithms produce the right kind of solution, or whether

![Figure 1. MEC task offloading example.](image-url)
there exist constraints we did not yet discover, for instance.

- Evolutionary algorithms outperform in the optimization problems which are directly and inherently suitable to deal with such optimization tasks, that is, tasks where we have to tradeoff between multiple, conflicting optimization goals, that is, energy and delay.

The process of making a distinction among tasks either to perform offload or local computation is considered as an optimization problem. For making this decision, minimization of energy consumption and response time are the two important objectives. However, these objectives have a conflict of concerns. Improvement in energy consumption occurs at the cost of extended response time and vice versa. Therefore, it is important to select the right mix of offloading and local computing tasks such that energy consumption and response time are minimum. The following are the main contributions of this article as follows:

- To minimize the energy consumption of the mobile devices and IoT.
- To minimize the response time for task computation at MEC servers.
- Optimally decide whether the task is to be computed locally on the mobile device/IoT or to be offloaded on the MEC server.

The remainder of this article is organized as follows. In section “Related work,” we present the related work. Section “Research methodology” describes the proposed methodology for the optimal decision of offloading in MEC systems using evolutionary algorithms. Section “Experimentation and result analysis” describes experiments and analysis of the results. In section “Tradeoff analysis between energy and delay,” we graphically denote the tradeoff between energy and delay, and then at the end, we conclude this article with future work.

Related work

From the last few years, the advancement in mobile devices and IoT has significantly changed, and there is an exponential growth in the Internet traffic in networking and wireless communication. In particular, the growth in multi-link Internet connections provides a high speed of gigabit wireless speed in the next-generation systems. MCC creates a huge latency and delay which is unbearable in real-time scenarios and in other cases as well. Presently, the main perspective of 5G is to provide the high-speed connection with low latency and high bandwidth with multi-channel communication so that we not only support the voice transition but also cope the multimedia-centric traffic. It is now agreed that we should not only rely on MCC to get low latency and a high data rate in the 5G concept. So, it is essential to make mobile edge servers with cloud servers for better computations and storage. MEC uses base station (BS) for offloading computations and storage. Cisco has also proposed terminology of “Fog Computing” in the same concept of the Edge Computing, and it will lead us to the new research area of Fog Computing and MEC.

To fully utilize the architecture of MEC systems, energy efficient, task scheduling, and offloading computation problems have been extensively studied. A technique is proposed with an energy competent binary offloading under a contingent wireless channel. In another technique, a computational resource and task scheduling allocation policy will be introduced in heterogeneous networks. Mao et al. proposed an algorithm which performs online offloading computation by cooperatively considering radio and computational resources management in multi-user MEC systems. The main design issues are when, what, how, and which task is to be offloaded on the MEC server and which tasks are to be serviced locally on the mobile device and IoT.

Recently, a new technique is proposed called the age of information (AoI) which computes the freshness of the task and information compared to the existing edge computing solutions. Yousafzai et al. analyzed the effects of platform-dependent applications on offloading computation in MEC and proposed a process of lightweight migration-based offloading task computational context. It also proposed a process migration-based computational offloading (PMCO) framework that basically migrates to resource-rich computing infrastructure from a resource limited mobile. Basically, they evaluate the proposed features and their lightweight feature using standardized using experiments keeping in mind the execution time of tasks, energy consumption, and amount of transferred data. With the expeditious advancement in the aggregation of IoT, mobile devices, and 5G networks in the smart cities, a large amount of big data is to be produced which consequences in the increased latency for the traditional MCC and MEC system. Placement of the edge servers is bit challenging task. The distance is to be optimized at which the edge server is to be placed to get the maximum benefits, and more devices have been connected to the server and perform the offloading tasks. Wang et al. and Xu et al. took the placement of edge server as the optimization problem and assumed that each server has the same computational power to compute and process mobile users and IoT tasks. The objective is to balance the load in such a way that the tasks are divided into equality to not overburden any server at any cost.
Offloading the task to the MEC servers absorbs more energy and latency, but on the other side, it is necessary to balance the network in a way that the resource blockage to the edge servers will be minimized on MEC. There is a compensation between the offloading scale and the quality of service (QoS). A unique three-layer integration architecture is introduced in Lyu et al. which includes the MEC, MCC, and IoT. Authors proposed a lightweight request and admission framework in which selective task offloading is evolved to minimize the energy consumption of the Mobile devices and IoT. With the metamorphic development of delay-sensitive applications, latency becomes a challenging task to run computationally intensive tasks and applications on mobile devices. In Ning et al., it starts from a single user offloading computation task problem, where the MEC resources are not considered. The branch and bound algorithm are used for this solution. Later, a problem is formulated where multi-user offloading computation is assorted with mixed integer linear programming (MILP) problem by taking into cogitation the resource competition between mobile users and IoT and considers this issue as an NP-hard problem. Due to this complexity, a repetitive heuristic MEC resource allocation (IHRA) algorithm is offered to design and make the dynamic offloading decision whether the task is to be computed locally on the mobile devices or on offloaded to the MEC server. The IHRA technique is acceptable for different kinds of applications. A framework is suggested and proposed in which it is developed to tackle packet delay, response time, and gain offload ratio for both MCC and MEC systems. A new technique is introduced to evaluate the performance of the MEC systems called the computation efficiency which is defined as the number of calculated data bits divided by the energy consumption accordingly.

The mobile user always desires performance guaranteed and low energy consumption. Tao et al. proposed an energy efficient optimization problem for MEC computing. The offloading computation activity where tasks are to be determined by bandwidth capacity and energy consumption at each time interval. This paper addresses the problem of the performance of the offloading computation scheme for MEC. It formulates an energy minimization problem with delay time and resource parameters. Another method is proposed to decide the offloading decision of the tasks. Offloading computation decisions are taken by observing the current energy conditions of mobile devices and the requirements of its application. The edge server decides on each coming mobile request based on their priorities and complete performance.

In Zhang et al., a distributed joint offloading computation and resource allocation optimization (JCORAO) technique is used in heterogeneous networks (HETNETS) with MEC is proposed. This optimization problem is divided into two subproblems. One for analyzing offloading strategy, an algorithm is used called distributed potential game is used. Another sub-algorithm named cloud and wireless resource allocation algorithm (CWRAA) is used which jointly allocates the uplink subchannel, uplink transmission power, and computation resource for the offloading tasks. Hence, an optimization model is proposed in Zhang et al. that reduces endurable delay mobile network (MTD) by considering both the jitter random average delay. The authors also proposed an efficient conventional mixed earliest-finish time (CHEFT) algorithm to solve the MTD-minimization problem. The main contribution of this paper is that MTD-minimization task scheduling problem in the MEC system. It considers both characteristics which are delay time and delay jitter.

Most of the work considers the selection of offloading as an optimization problem. However, none of them used evolutionary algorithms of artificial intelligence to find optimal results. To perform the selective offloading with multiple input tasks such that the overall network energy consumption of IoT and the overall network delay become minimum. Evolutionary algorithms have the capability to explore and exploit the search space and get the best results.

Research methodology

This section describes the proposed framework in detail. We follow the MEC system architecture shown in Figure 2. Table 1 enlists the notations and their meanings used throughout the article.

![Figure 2. MEC task offloading architecture.](image-url)
System model

An edge server relates to multiple IoT via BS. To keep the edge server updated with mobile devices, periodic messages are exchanged. These messages allow the edge server to have complete knowledge of connected devices, and their parameters, that is, $F_m$, $P_t$, and available bandwidth $B$.

During operational life, different IoT/mobile devices may perform variety of tasks ranging from face recognition to intelligent transportation system. Some of these tasks are very simple in nature and need few CPU cycles for execution, whereas others require extensive computations.

Figure 2 illustrates the decision-making process of MEC system. Whenever a device is triggered to perform a certain task, it senses the input; calculates $d$, $C$ to compute that task and $\tau$ for delay-sensitive tasks; and sends them to edge server. The edge server directs the received $(d, C, \tau)$ and pre-fetched $(F_m, P_t, \text{and } B)$ parameters to the MEC controller, which in turn runs the evolutionary algorithms to take offloading decision.

Optimization problem formulation

The optimal selection leads to minimize the response time and energy consumption of IoT devices which can easily be defined mathematically in equations (1) and (2) as follows

$$T_m > T_{off} \quad (1)$$

and

$$E_m > E_{off} \quad (2)$$

where $T_m$ and $T_{off}$ are the time to compute the task locally at the mobile device and at MEC server, respectively. $E_m$ and $E_{off}$ are the energy consumption to execute tasks locally at the mobile devices and at MEC server, respectively. It is evident from equations (1) and (2) that it is better to execute a task locally at mobile device if $T_m$ and $E_m$ are less than $T_{off}$ and $E_{off}$, respectively. This optimal decision is based on the amount of energy and time required to compute the task. The mobile devices send the task-related parameters such as task size, node capacity, and network condition to the edge server. The controller on the server runs an evolutionary algorithm and decides the tasks which are to be executed locally on the mobile device or send for offload computation. Equations (1) and (2) state that offloading a task is only beneficial if $T_m$ is greater than $T_{off}$ and $E_m$ is greater than $E_{off}$. $T_{off}$, $T_m$, $E_{off}$, and $E_{off}$ can be calculated using equations (3)–(6), respectively

$$T_m = \frac{dC}{F_m} \quad (3)$$

$$T_{off} = \frac{d}{B} + \frac{dC}{F_t} + T_{wait} \quad (4)$$

$$E_m = \alpha CdF_m^2 \quad (5)$$

and

$$E_{off} = P_t \frac{d}{B} + E_{idle} \quad (6)$$

In addition to the processing delay at the server, $T_{off}$ also includes the waiting time $T_{wait}$ at server and transmission time for sending data to the server. It is apparent from equations (3) and (4) that if the “$d$” (input data) is small and required large number of “$C$” for executing the task (i.e. computationally intensive task), then it is better to offload the task at MEC server. Consequently, a task with large value of “$d$” and small number of “$C$” takes enough time for offloading it to the server, and therefore, it is better to compute the task locally at device. In the same way, a heavy loaded server having many processes waiting in the buffer forces the task for local execution considering the computation cycles and data size of the task.

Some of the IoT have enough computational resources and use their full capacity to execute the task within a given time frame. However, this reduction in $T_m$ occurs at the cost of extra depletion of battery energy since $E_m$ is directly proportional to the square of CPU cycle utilization as shown in equation (5). $\alpha$ is the architectural constant of CPU. $E_{off}$ consists of energy consumed while offloading the task, and during the transmission of data to the server. It also includes energy leakage when the device is idle during offload execution. Equations (4) and (6) clearly explain that if the network conditions are not favorable (i.e. provide
low operating bandwidth), it is better to execute the task at mobile device since transmission time becomes the prominent factor. The edge server receives the parameters from all nodes in responsibility and takes network-wide decisions. It is important to mention that the server does not only decide for a single node and enhance its efficiency rather it aims to optimize the performance of the whole network and take accumulative decisions. Therefore, it is mandatory to consider the energy consumption and processing delay for the whole network.

Let us assume that edge server has perfect knowledge of $P_r$, $F_m$, $w$, and $B$. The server must take optimal decision and select only those tasks for offload computation such that sum of network energy consumption and sum of network latency are minimized. Since, reducing the response time and saving the nodes energy are minimization nature problems, equations (7) and (8) define the objective function for response time and energy consumption, respectively.

Objective 1

$$f(T) = \sum_{i=1}^{k} \left( \frac{d_i}{B_i} + \frac{d_iC_i}{F_m} \right) + \sum_{j=1}^{l} d_jC_j \quad \forall = k + l \quad (7)$$

Objective 2

$$f(E) = \sum_{i=1}^{k} \left( \frac{P_{r,i}w}{B_i} + E_{idle,i} \right) + \sum_{j=1}^{l} \alpha C_j d_j F_m^2 \quad (8)$$

where $f(T)$ and $f(E)$ are the objective functions for time and energy, respectively. We used the basic concept of optimization in these algorithms and design our new variant design to solve this problem through mathematical modeling.

**Evolutionary algorithms**

Evolutionary algorithms are well-known for solving optimization problems. They create the search space of multiple solutions where each solution consists of all feasible elements that satisfy certain criteria. The worth of each solution is evaluated against fitness function. Higher the fitness value of a solution, better the solution will be. The fitness of the solutions is used to opt for the best solution from the whole search space. Each solution is then updated based on the distance from the best solution. Algorithm 1 outlines the procedure of the proposed model with evolutionary algorithms. The input data consist of task size, the processing power of the mobile device, available bandwidth, and residual energy. Input data are given to the MEC server allocates the optimization job to MEC controller.

![Figure 3. Flow chart.](image)

In figure 3, it is the MEC controller who runs the evolutionary algorithm. The detailed description of the prominent function is given in subsequent sections.

**Create search space**

Search space of $nPop$ number of solutions is created where $nPop$ is the population size. Each solution has $N$
number of elements where each element corresponds to the ID of the mobile devices. Since selecting the task for offload computation is the binary nature problem (offload or local computation), the element of the solution contains the binary value, that is, 1 or 0. Figure 4 visualizes the structure of the solution. The element value 1 or 0 denotes the offload or local computation of task, respectively.

To generate a solution, random selection is made from set of available tasks for adding first element to the solution. The subsequent elements are added into the solution only when they satisfy the following conditions:

1. If it is not already present in the solution.
2. Computation of sum of the selected tasks is less than the server available capacity, that is, $F_s = \sum_{i=1}^{k} d_i C_i$.
3. $T_m + E_m > T_{off} + E_{off}$.

The third condition dictates that only those tasks will be selected for offload computation that requires a significant amount of computation. It is better to execute small tasks locally at mobile device since they produce required results within the given time frame without wasting device resources.

### Fitness function

Fitness function (also known as cost function) is one of the critical parts of any algorithm. The accuracy of an algorithm is determined by the fitness function. A well-defined fitness function will lead to the attainment of optimal results. Accurate results can only be produced by removing the biasness of parameters in the fitness function. It must be aligned with objectives that are defined to solve the stated problem.

In the given problem of selecting tasks for offload computation, the main objectives are to minimize the response time and save the nodes energy. Therefore, the fitness function will be

$$f = w_1 f(T) + w_2 f(E)$$

where $f(T)$ and $f(E)$ can be computed using equations (7) and (8), respectively. $w_1$ and $w_2$ are the user defined parameters and associated with time and energy, respectively. These are used to intensify the importance of delay critical or energy sensitive applications. A higher value of $w_1$ means response must be given within the bounded delay, whereas $w_2$ is set higher for applications that require long operational life of device.

### Stopping criteria

Stopping criteria define the end of the execution of evolutionary algorithms. In the proposed model, we used two stop criteria for controlling the execution. First, the algorithm stops when there is no improvement in best solution found so far. If there is no more improvement in the classification of the best and optimal results, then the iteration is called the Stall iteration. This situation occurs when the algorithm has converged and found the best solution. The algorithm also stops when the limit of a total number of iterations is reached. After stopping the execution of algorithm, the best trail found so far is used for offloading nodes.

### Experimentation and result analysis

In this section, performance comparison of task optimization models is carried out. We used ACO, GWO, and WOA, and then compare their results to get the optimized solutions for offloading strategy. Experiments are performed on MATLAB, and the detail of simulation parameters is given in Table 2. In this article, the face recognition algorithm as described in Soyata et al.\(^2\) has been used as a computation task. According to Table 1, MEC server is used which has 20 GHz computing power. The communication bandwidth between the mobile nodes and the MEC server is set to 10 MB. We categorized the total number of nodes in a way that the computations will be performed when the total nodes will be set to 20, 40, 60, 80, and 100 to compare...
the results in all conditions of nodes. The task size of the data is distributed in three categories as small tasks (1 KB), medium tasks (500 KB), and big tasks (1000 KB). The processing power of the mobile devices/nodes is set to 0.6–1 GHz random distribution. The initial energy of the nodes is also set randomly ranging from 0.9 to 1.1 W. Transmission power is also defined as 20–23 dBm randomly.

Performance of ACO, GWO, and WOA is evaluated in terms of network energy consumption, the sum of delay, and the number of tasks offloaded. The value of $w_1$ and $w_2$ is set to 0.5 to give equal importance to energy and delay. We use an average of delay suffered and energy consumption of all nodes in the network. The objective of taking these network-wide parameters is that we want to optimize the energy and delay for the whole of the network and not only for a single node. Although it might be possible that offloading a task of any node is beneficial for the network, it may hamper the performance of other nodes.

Network energy consumption

Energy is one of the scarcest resources of the IoT. The operational life of an IoT is heavily dependent on the pattern of energy consumption. Figure 5 shows the energy consumption for small, medium, and large tasks, respectively. It can be seen from the figure that GWO outperforms the ACO and WOA. GWO performs the optimal selection of offloading tasks. It only selects the task for offload computation which results in saving energy of all nodes. The figures also depict that increasing the task size (whether it is executed locally or offloaded) increases energy consumption. Moreover, with the increase in the number of tasks, energy consumption also increases. Both factors lead to an increase in the sum of computation load on either IoT or edge server. If the computation is performed locally, energy consumption will be due to CPU cycle utilization. Alternatively, energy consumption for an offloaded task is incurred due to the transmission of data and reception of results to/from the edge server.

Computation latency

Computation latency (also known as delay) is the time elapsed since the IoT senses/takes input and produces output. It is execution time that can be computed using equations (3) and (4) for local computations and offload computations, respectively. The shorter the value of delay, the better will be the user satisfaction. We have already discussed earlier that the real-time applications require minimum delay, that is, face recognition system. The requirements of such real-time application are increasing with the trend of emerging science and in the field of IoT. We perform experiments on three algorithms to get the optimal decision where we finally get the minimum delay to compute the given task. Delay comparison of ACO, GWO, and WOA is shown in Figure 6 for small, medium, and large tasks, respectively. Same as for energy consumption, computation latency becomes prolonged with the increase in size and the number of tasks. Hence, it is proved that the performance of GWO is better in all these instances and algorithms.

Number of offloading tasks

Offloading tasks are those tasks that are being sent to the edge server for offload execution. Finding an optimal set of offloading tasks is crucial. Sending all tasks for offload computation will overburden the edge server and results in additional delays. However, executing all tasks at mobile devices will deplete their energy. As it has already been described that a task is only sent for offloaded computation if it requires a larger number of CPU cycles for execution. Larger the CPU cycles required; the higher will be the probability for offload execution. In contrary to this, small tasks are required
to be executed locally on mobile devices to save the energy, bandwidth, and server utilization. Moreover, the tasks having larger input data size and require a smaller number of CPU cycles for per bit computation are also preferred for local computation because they consume a significant amount of energy and time during transmission of data to edge server. Figure 7 shows the number of tasks offloaded with various executions at an instance of time. GWO selects the optimal of offloading tasks which helps in saving energy and reducing the execution time of all available tasks. Increasing the number of tasks in hand enforces to increase the number of offloaded tasks. Degrees of task size and the number of tasks available. However, after reaching saturation point (i.e. when server execution has reached its capacity), the ratio of offloading to local computing declines.

Tradeoff analysis between energy and delay

Executing a very small task (1 KB) over an MEC server causes increased energy consumption due to communication and executing multiple tasks at the MEC server tends to provide high response time that leads to overloading and put additional latency at task computation. Moreover, if the task size is too small, executing it over an edge server causes increased energy consumption due to communication. For delay-sensitive applications where IoT has limited battery energy. Therefore, it is required to have an optimal selection for tasks offloading such that the response time and energy consumption will become minimum. We graphically represent the two graphs in Figure 8. One graph has energy-delay tradeoff with small task size (1 KB) and the other shows results for energy-delay tradeoff for large task size (1000 KB). It is clearly depicted that energy and delay has an inverse relationship. Decreasing the energy consumption tends to increase the delay increases. Similarly, a quick response can be achieved at the cost of depleting extra energy. A tradeoff between delay and energy will ultimately help to choose an optimal point (Pareto optimal solution) where we can save enough energy with minimum delay.

Conclusion and future work

As a fact that the limited energy and computation power of mobile devices are unable to meet the needs of computationally intensive tasks. In this article, we proposed the meta-heuristic-based task offloading model.
for the MEC system. In the proposed model, we aimed to optimize energy consumption and minimize execution latency while selecting the right mix of offloading tasks. Simulation results show that the performance of GWO is relatively much better than ACO and WOA. The lead in performance is because of the optimal selection of a set of offloading tasks. However, the evolutionary nature of these metaheuristics abides to take too much time while taking offloading decisions, which is not realistic for real-time systems. We also graphically elaborate the tradeoff analysis between energy and delay for small task size and large task size for all three evolutionary algorithms. It has been noticed that the energy and delay have inverse relationship. In the future, we aim to identify the actual reasons for this long duration and attempt to minimize it. Moreover, a threshold can be defined for small tasks to reduce the control communication which is being held between edge server and mobile device.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD
Farhan Aadil https://orcid.org/0000-0001-8737-2154

References
1. Khan AR, Othman M, Madani SA, et al. A survey of mobile cloud computing application models. IEEE Commun Surv Tut 2014; 16: 393–413.
2. Fahad M, Aadil F, Khan S, et al. Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks. Comput Electr Eng 2018; 70: 853–870.
3. Shakarami A, Ghoubei-Arani M and Shahidinejad A. A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective. Comput Netw 2020; 182: 107496.
4. Shakarami A, Shahidinejad A and Ghoubei-Arani M. A review on the computation offloading approaches in mobile edge computing: a game-theoretic perspective. Software Pract Exp 2020; 50: 1719–1759.
5. Mann ZA. Optimization problems in fog and edge computing. In: Buyya R and Srirama SN (eds) Fog and edge computing: principles and paradigms. New York: IEEE, 2019, pp.103–121.
6. Chiang M and Zhang T. Fog and IoT: an overview of research opportunities. IEEE Internet Things 2016; 3: 854–864.
7. Klas GI. Fog computing and mobile edge cloud gain momentum open fog consortium, ESTI MEC and cloudlets. 2015, https://yucianga.info/wp-content/uploads/2015/11/11_11_22_Fog_computing_and_mobile_edge_cloud_gain_momentum_Open_Fog_Consortium-ESTI_MEC-Cloudlets_v1_1.pdf
8. Zhao T, Zhou S, Guo X, et al. Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In: Proceedings of the 2017 IEEE international conference on communications (ICC), Paris, 21–25 May 2017, pp.1–7. New York: IEEE.
9. Mao Y, Zhang J, Song SH, et al. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE T Wirel Commun 2017; 16: 5994–6009.
10. Song X, Qin X, Tao Y, et al. Age based task scheduling and computation offloading in mobile-edge computing systems. In: Proceedings of the 2019 IEEE wireless communications and networking conference workshop (WCNCW), Marrakesh, Morocco, 15–18 April 2019, pp.1–6. New York: IEEE.
11. Yousafzai A, Yaqoob I, Imran M, et al. Process migration-based computational offloading framework for IoT-supported mobile edge/cloud computing. IEEE Internet Things 2020; 7: 4171–4182.
12. Huang L, Zhang L, Yang S, et al. Meta-learning based dynamic computation task offloading for mobile edge computing networks. IEEE Commun Lett 2021; 25: 1568–1572.
13. Wang S, Zhao Y, Xu J, et al. Edge server placement in mobile edge computing. J Parallel Distri Com 2019; 127: 160–168.
14. Xu Y, Wang Y and Yang J. Meta-heuristic search based model for task offloading and time allocation in mobile edge computing. In: Proceedings of the 2020 6th international conference on computing and artificial intelligence, Tianjin, China, 23–26 April 2020, pp.117–121. New York: ACM.

15. Chen X, Jiao L, Li W, et al. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE ACM T Network 2016; 24: 2795–2808.

16. Lyu X, Tian H, Jiang L, et al. Selective offloading in mobile edge computing for the green Internet of Things. IEEE Network 2018; 32: 54–60.

17. Ning Z, Dong P, Kong X, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. IEEE Internet Things 2019; 6: 4804–4814.

18. Rimal BP, Van DP and Maier M. Mobile-edge computing versus centralized cloud computing over a converged FiWi access network. IEEE T Netw Serv Man 2017; 14: 498–513.

19. Sun H, Zhou F and Hu RQ. Joint offloading and computation energy efficiency maximization in a mobile edge computing system. IEEE T Veh Technol 2019; 68: 3052–3056.

20. Tao X, Ota K, Dong M, et al. Performance guaranteed computation offloading for mobile-edge cloud computing. IEEE Wirel Commun Le 2017; 6: 774–777.

21. Zhang J, Xia W, Yan F, et al. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. IEEE Access 2018; 6: 19324–19337.

22. Zhang W, Zhang Z, Zeadally S, et al. Efficient task scheduling with stochastic delay cost in mobile edge computing. IEEE Commun Lett 2019; 23: 4–7.

23. Subramaniam EVD and Krishnasamy V. Energy aware smartphone tasks offloading to the cloud using gray wolf optimization. J Amb Intel Hum Comp 2021; 12: 3979–3987.

24. Zakaryia SA, Ahmed SA and Hussein MK. Evolutionary offloading in an edge environment. Egypt Inf J. Epub ahead of print 9 October 2020. DOI: 10.1016/j.eij.2020.09.003.

25. Soyata T, Muraleedharan R, Funai C, et al. Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: Proceedings of the 2012 IEEE symposium on computers and communications (ISCC), Cappadocia, 1–4 July 2012, pp.59–66. New York: IEEE.