Energy-Aware Computation Offloading in Mobile Edge Computing Using Quantum-Based Arithmetic Optimization Algorithm

Mohammad Masdari
Islamic Azad University
Kambiz Majidzadeh
Islamic Azad University
Elahe Doustsadigh (Edoostsadigh@gmail.com)
Islamic Azad University
Amin Babazadeh
Islamic Azad University
Reza Asemi
Islamic Azad University

Research Article

Keywords: Offloading, Mobile Edge Computing, Arithmetic Algorithm, Quantum, Energy consumption

Posted Date: November 3rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2221212/v1

Additional Declarations: No competing interests reported.

Energy-Aware Computation Offloading in Mobile Edge Computing Using Quantum-Based Arithmetic Optimization Algorithm

Mohammad Masdari[1*], Kambiz Majidzadeh[2], Elahe Doustsadigh[3], Amin Babazadeh[4], Reza Asemi[5]
Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
M.masdari@Iaurmia.ac.ir[1] , Kambiz.majidzadeh@iau.ac.ir[2] , Edoostsadigh@gmail.com[3] ,
bsamin2@liveutm.onmicrosoft.com[4], r.asemi86@yahoo.com[5]

Abstract: The Internet of Things (IoT) has rapidly grown recently, and mobile devices (MDs) have encountered widespread usage. All of these cause an increase in the demand for more powerful computing resources. Meanwhile, a new concept called mobile edge computing (MEC) was introduced as a promising technology to access powerful computing resources closer to the user side for a quick and effective response, especially for time-intensive applications. Task offloading has emerged as a solution to allocate resources among computing resources of smart devices or computational resources available in MEC. This study presents a new binary quantum approach based on an arithmetic optimization algorithm (BQAOA) for computational tasks offloading decisions on MDs with low complexity and guaranteed convergence. However, since task offloading is an NP-hard problem, there is a need to use methods that provide the optimal possible solution for various quality criteria, including response time and energy consumption. Indeed, this is where the advantages of arithmetic optimization algorithms (AOA) and quantum computing have been used to improve the performance of MDs. This paper introduces a 2-tier architecture from the user to the cloud computing server-side. Also, a Markov model is proposed to compute the average network bandwidth in the offloading problem. The proposed BQAOA is compared with the best state-of-the-art algorithms in heuristic and meta-heuristic fields in different scenarios. The simulation results showed 12.5%, 12%, and 26% improvement in energy consumption, makespan, and Energy SLA Violations (ESV) optimization parameters, respectively.

Keywords: Offloading, Mobile Edge Computing, Arithmetic Algorithm, Quantum, Energy consumption.

1. Introduction

According to one definition, the IoT is a collection of interconnected devices, from simple sensors to smartphones and wearable things. Devices on private or closed networks may interact with one another or the outside world using the IoT. IoT empowers a more interconnected world since it allows gadgets to interact inside their own silos and across other networking forms. In this scenario, as more and more sensors and devices are networked through IoT approaches, the data generated by these sensors and devices will become enormous, necessitating further processing [1]. Moreover, because the bulk of IoT devices has limited power, it is vital to balance power usage by allocating computation tasks to devices with more sizable computing capability. It is also possible to decrease transmission time and power consumption expenses by processing data in computing nodes most distant from the consumer [2-4].

It may sound necessary to offload apps needing extensive processing resources to a traditional centralized cloud to improve the battery life of the MDs. Although this alternative is more cost-effective, it results in a considerable execution delay due to the time spent moving apps to and from the cloud and the time spent on cloud computing. Offloading is unsuitable for real-time applications because of its latency. A newly developed concept known as MEC has arrived to deal with the delay issue. The MEC offers a variety of resources. Here, computation and storage resources are brought to the edge of the mobile network. This new system will allow for the execution of the most demanding apps on the MDs while adhering to stringent time-sensitive criteria [5]. MEC uses a smaller portion of the network for its traffic. Thus, it can reduce network congestion. Also, MEC can be used in intelligent manufacturing for local data analysis and storage and offers more stable connections [6]. After all, there is less competition for the link to the edge node. Due to the limitation of computational resources on edge and the possibility of implementing the user's favorite computational application with minimum energy consumption and quick processing time, increasing performance requires several policies [7]. One such policy is offloading to utilize the computation concept. As developing recent applications require excessive computing power, there is a need to offload computational tasks from intelligent devices to the edge layers and ultimately to cloud servers to increase the processing power of IoT devices. This offloading, by nature, equal the practical implementation of application and efficient energy consumption on the user side [8]. As a result, applications with high processing requirements can run on the user-side edge computing by placing IT resources closer to the end-user and outside a central data center or cloud environment. MEC and 5G are tied together because the small cell deployments for urban environments are planned to have computing capabilities [9].

Offloading solutions should address some issues to be effective in the MEC setting. The first issue relates to the wireless channels, intermittent cloudlets, and cloud servers used for fault tolerance in offloading. As a result of these mechanisms, handling offloading failures is critical. The second issue is protecting user data during transmission and at the destination nodes not under the users' control because of its transference to remote cloud servers. Third, since the offloading process might change depending on a user's location and context, an MD should evaluate and infer the context information. Even when utilizing dynamic decision-making programs, the tasks that may be delegated are identified during the development process. However, due to the high analysis costs, partitioning is undesirable during program execution.

Offloading strategies can be categorized into two groups:

- **Complete offloading:** When a program is offloaded entirely, it goes to an offloading server rather than staying on an MD. However, if the program's size is more than the combined size of its components, it may result in significant network overhead.
- **Partial offloading:** A section of a program or workflow will be offloaded when using partial offloading. The sMEC server interaction overhead should be well-handled since it might offer a runtime overhead for a client program.

In various cases, offloading systems can also use 3G, 4G, and 5G cellular networks. Additionally, channel availability might be continuous or sporadic. Power management in offloading methods might be static, dynamic, or renewable. Using techniques like DVFS, dynamic power management offloading systems seek to minimize energy usage on both the MDs and destination sides. Although minimizing energy usage, the MDs or destination nodes will still fulfill QoS standards. Offloading solutions may also explore using renewable energy at the destination sites, such as cloud DCs. MDs can also use cloudlets, fog computing, or MCC to offload queries. The choice to unload might be taken locally or globally.

Computational offloading is an NP-hard problem because of the complexity of resource allocation and discrete offloading decisions [10-12]. Many offloading schemes for MEC have been proposed using different methods like stochastic or meta-heuristic algorithms to deal with the offloading problems [13-15]. A meta-heuristic algorithmic framework is a high-level, problem-independent algorithmic framework that offers a collection of principles or methods for developing heuristic optimization algorithms. Some examples of meta-heuristic algorithms include genetic/evolutionary algorithms [16], Tabu search [17], simulated annealing [18], variable neighborhood search[19], ant colony optimization [20], among others. However, many others exist as well. In general, meta-heuristic algorithms may be divided into the following categories [21]:

- Local search,
- Constructive,
- Population-based, and
- Hybrid.

In this paper, a new meta-heuristic method called the AOA is utilized. This method, among others, takes advantage of the distribution behavior of the four principal arithmetic operators in mathematics: multiplication (M), division (D), subtraction (S), and addition (A). To conduct the optimization procedures in extensive search areas, AOA is theoretically modeled and implemented in a computer program [22].

This paper presented BQAOA, a new binary quantum-based AOA algorithm to deal with offloading problems. The BQAOA is based on AOA Approach to find the best method using quantum algorithms exploiting quantum entanglement; the offloading decision problem can be solved more efficiently than on classical peers. The rationale for the binary version of BQAOA is that the final solution is a matrix problem of zeros and ones. Then, BQAOA can solve NP problems with considerable time complexity to increase convergence and reduce constraints. Furthermore, the resulting BQAOA approach's energy consumption was reduced while maintaining the performance. Next, the proposed BQAOA algorithm is utilized to optimize solutions with acceptable QoS parameters. Finally, the proposed algorithm is compared with other state-of-the-art algorithms using various metrics to show its superiority.

The main contributions of this paper are as follows:

- A binary version is provided for the AOA algorithm to solve the offloading problem in MEC Systems within the first stage; a solution had to be proposed for the AOA algorithm to work discretely. The Sigmund function has been used in this research to transform the problem space of the AOA algorithm from continuous to discrete.

- The Quantum strategy was embedded in BQAOA to select the appropriate solution from the problem space. This study attempted to provide a quantum version of the binary AOA algorithm using quantum computational theory to enhance the overall performance of the binary AOA algorithm. The purpose was to converge binary AOA quicker and help escape local optimization, and ultimately the proposed BQAOA algorithm was named.
- Balanced and efficiently distributed the processing load at exclusive processing levels. In this research, through the BQAOA algorithm, the processing load has been attempted to distribute the tasks in a balanced way at different levels: the lowest level of MDs and the best level of effective mMEC servers. The proposed solution attempts to distribute the tasks the usage of the using the BQAOA algorithm such that the total energy consumed in the mobile layer of devices, sBEC servers, and mMEC servers is minimal. It additionally attempts to keep the makespan in optimal condition.
- Since the bandwidth is variable between IoT and MEC, a Markov model is considered to achieve an optimal average bandwidth. This model can prevent over-estimating the network bandwidth and improves the effectiveness of the proposed scheme.
- A comprehensive set of simulations are conducted to evaluate the performance of the proposed scheme regarding different metrics.

This article is organized as follows: The previous works are described in section 2. In Section 3, the AOA algorithm is presented. In Section 4, the problem with our proposed binary quantum AOA-based algorithm is formulated to increase the performance of MDs and decrease the completion time and the total energy consumption. Section 5 describes the BQAOA algorithm. The proposed offloading scheme is described in section 6. The performance evaluation and the numerical results are presented in Section 7. Finally, Section 8 concludes the paper.

2. Related works

In [23], Sun et al. introduced a resource scheduling method based on a non-dominated sorting genetic algorithm (NSGA-II). The scheduling model has two-level performing scheduling among fog clusters as first-level performing scheduling in the same fog cluster during the fog nodes. This approach helps service latency and improves stability.
Also, in[24], with emphasizing this fact that processing time is one of the vital factors in the IoT devices as QoS parameters and with relying on the advantage of fog computing, which is one of the suitable options to solve the scheduling problem in the IoT, two meta-heuristic algorithms, ACO and PSO are proposed to solve the problem of scheduling computational tasks to be able to reduce response time and achieve a significant improvement in load balancing. This approach helps improvement in response times and effective balances.
In [5], Bitam et al. presented a task scheduling scheme for fog computing that achieves less execution time and meets the service computing needs of mobile users. It applies an optimization algorithm based on the bees life algorithm and can reduce CPU execution time and the total amount of memory. This approach helps in execution time and allocated memory. In [25], Du et al. provided an exploration on the interaction of cloud and fog in mobile computing to improve the issue of loading services for smart devices that have inherently computational tasks. In fact, by proposing a mixed cloud/fog model that can take advantage of both loading decisions and the issue of resource allocation in cloud computing. A low-complexity sub-optimal algorithm is proposed where the semi-definite relaxation and randomization implement offloading decisions, and also fractional programming theory and Lagrangian dual decomposition obtain resource allocation. The benefit of this approach is cost conversation.
In [26], Abro et al. introduced a solution to deal with task scheduling in fog computation. The goal is to place local and remote tasks on VMs in a fog environment, so the total energy consumption is combined with local energy fog resources energy consumption. The joint energy-efficient task assignment (JEETA) is proposed on a new fog cloud architecture that performs a novel algorithm, the dynamic application-partitioning algorithm (DAPTS), to decide the offloading decision for achieving better results in energy consumption. This approach helps in remote task performance and QoS. In [27], Gharehpasha et al. presented a new method based on a hybrid discrete multi-object whale algorithm with a combination of chaotic functions to optimal offload decision in the cloud environment. This study helps in scaling decisions.
In [28], Hazra et al. introduced a vision that the advent of MEC has reduced the latency of cloud services, as well as the emergence of a term called cloudlet, which is a small scale of cloud infrastructure in the edge of the cloud environment and the end-users side. The authors also refer that processing cloudlets on access points is costly, and even the service provider may not be able to guarantee a delayed latency. So with this point, out that the problem is NP-hard, a new method based on the capability of Mixed Integer Linear Programming (MILP) and Software-Defined Network (SDN)-based framework and a bender decomposition-based algorithm are proposed. This study helps in energy consumption and acceptance ratio.

Table 1.The properties of the offloading methods

Ref	Algorithm/method	Environment			Simulation Metrics	Shortcoming
		Fog	Edge Computing	MEC		
[23]	Genetic Algorithm	✓			Service latency, stability	It is not economical in terms of cost
[24]	Ant Colony, PSO	✓			Response time, Balance	It is not economical in terms of cost and tasks dependencies
[5]	bees swarm	✓			Execution time, memory.	Does not support dynamic jobs
[25]	CORA, BCRA	✓			Energy, Delay	The access period is not considered
[26]	A dynamic application partitioning algorithm	✓			QoS, Energy	It is not economical in terms of response time and energy
[28]	Benders decomposition-based		✓		Energy consumption	The queuing is not considered
[29]	HOM		✓		Total latency	It is not economical in terms of cost
[30]	The heuristic approach		✓		Energy consumption	It is not economical in terms of time
[27]	A machine learning-based		✓		Time, CPU	The accuracy is low
[31]	The heuristic approach		✓		Overhead	It is not considered hierarchy
[32]	JROPSO algorithm		✓		Time, energy	It is not compared to known methods
[33]	greedy hill-climbing			✓	Energy, time	The scalability is not considered
[34]	MinHop, METComm, MCTComm, MinMinComm, MaxMinComm, SufferageComm			✓	Makespan, Waiting time,	It is not economical in terms of energy
[35]	Particle Swarm Optimization			✓	Time	The live migration is not considered
[36]	Gray wolf optimization			✓	Energy	The probability and real-time are not considered
[37]	Ant-Bee Algorithm			✓	Energy	Online scheduling is not considered.
[38]	(ACS-JS)			✓	Speed, Bandwidth, Execution time	The workflow is not considered
[22]	ACO & CMSACO			✓	Time, Energy	Only compared with the static algorithms.
[39]	GKS algorithm			✓	Energy, Execution cost	It is not economical in terms of cost

In [29], XUet al. provided an offloading method for edge computing to process and complete the tasks at the cloud edge without sending them to the cloud core. This scheme applies deep learning and tries to reduce latency. This approach's advantage is to shorten the time delay. The offloading approach presented in [30] provides a hybrid model of multiple servers MECs and multiple MDs. Tasks can be run on either MDs or one of the MECs servers. This scheme tries to allocate tasks optimally and reduces their completion time. Also, a distributed heuristic algorithm to decide if the task is executed in the same MD and the processing power is fast, so a small number of tasks will be offloaded. Also, with the hope that the speed of communication links in 5G is getting higher, it can play an essential role in the decision to transfer tasks from MD to MECs. The benefit of this study is random probabilities.
In [31, 40], Pham et al. presented a framework for the allocation of computational and communication resources. The goal is to reduce computational overhead on multiple MEC Het-Net servers. A proposed method that has two parts is

the proposed computational loading decision and resource allocation connection. Two algorithms are considered. A new method to find placement transfer power for users is based on the duality method and the intercellular approximation method. The goal is to find the best processing resources for optimization. The advantage is the small optimality gap. The offloading scheme presented in [32] applies the PSO algorithm for making optimal decisions about offloading end-user applications in a multi-architecture, using multiple MEC servers in a 5G network. In the proposed method, an attempt has been made to optimize the energy consumption of end-users. In this case, if the tasks are executable on the end-user side, the task is processed on the end-user side. Otherwise, it is sent to the nearest server close to the user range. If a large processing volume is required, the task is sent to the cloud core. The advantage is high performance.
Enzai et al. [33],provided a new multi-site problem for mobile computation offloading, and a scheduler based on greedy hill-climbing is proposed for computational tasks both in cloud providers and end-users with satisfying energy consumption, task completion time, and charged prices. This study helps in good quality and achieving to a reasonable time. In [34], Li et al. introduced a decision problem in mobile ad hoc cloud-based networks on a set of combination batch and online scheduling heuristics methods. The idea is to find the best, more powerful cloudlet nodes. The six heuristics are proposed to respect the user and system parameters, such as load balancing and waiting time. The advantage is high performance in the online model. Guan et al. [35], presented a proactive cloudlet-based hybrid offloading model based on PSO to serve the energy and execution time efficiency. The time series-based prediction components are integrated to achieve proactive resource allocation.
In [36, 41], Veerappa et al. provided a multi-objective heterogeneous framework to increase the energy consumption of MDs. A new meta-heuristic algorithm task offloading method based on Gray Wolf Optimization (GWO) is described for offloading optimization problem. It reduces the cost of energy consumption between MDs and cloud servers. The study helps in the execution of energy in an efficient mode. The advantages are energy cost and high accuracy. In [38], Science et al. presented a queue-based and hybrid Ant Colony-Artificial Bee Colony Optimization (Ant-Bee) algorithm for task offloading in a Mobile Cloud Computing. The major purpose is to find an accurate place in the cloud or cloudlet. The combination of ACO with the 'Queue Decision Generator' implements the optimal assignment of a task. The advantages are high performance in completion time and power consumption.
Arunet al. [22] presented new research in mobile computing offloading for choose computing services such as computing time, energy consumption, and cost of using computing services for users' mobile tasks. A heuristic algorithm known as the accelerated cuckoo search algorithm is proposed to share resources to access high transfer rates in mobile processing computing. The study helps in multi-mode cloudlet performance. The [41], Guo provided a thesis on mobile cloud computing with the premise that mobile cloud computing is a predictive paradigm to address the growing demands of mobile users as users' processing needs increase as well as the number of sensitive computational applications. The delay is increasing, then the solution ILP and an efficient heuristic algorithm are proposed for offloading optimization problem. Experimental results have shown that the proposed method has achieved acceptable profitability in scalable problems. In paper [37], the Cooperative Multi-tasks scheduling algorithm based on Ant Colony Optimization algorithm (CMSACO) is proposed to solve the computational-intensive task for the offloading problem. This study helps in energy consumption, load balance, and channel state.
In [39], Wang et al. presented two sub-problems of optimization based on stochastic optimization methods to reduce the energy consumption of mobile users. Online task offloading is used to solve the problem, and the frequency scaling for Energy Efficiency (TOFFEE) algorithm is proposed to solve the second sub-problem. To achieve a reduction in power consumption, the authors have used both task allocation and CPU-cycle frequency. The advantage is high performance in energy consumption. Jeong et al.[42], presented that computation-intensive deep neural network (DNN) on MDs, so DNN requests are sent to the cloud core. Given that this is costly and time-consuming, trying to reduce time using the proposed Incremental Offloading of Neural Network (IONN) method based on the partitioning DNN offloading method. The IONN divides the user's DNN model into some partitions and then uploads them one by one. A graph-based algorithm is also used to select the best partition. This study helps in query performance and energy consumption.

3. The arithmetic optimization algorithm

In this study, a new meta-heuristic method called AOA is used that utilizes the main arithmetic operators in mathematics[22]. These arithmetic operators are:

- Multiplication (M " × "),
- Division (D " ÷ "),
- Subtraction (S " - ")
- Addition (A " + "))

Considering the population-based algorithms processes begin with a set of stochastic candidate solutions, the AOA starts the optimization process generating initial stochastic solutions in a matrix [43]. In each iteration, the best candidate solution or early optimum so far is considered. This algorithm consists of two phases based on the population-based optimization method process. Therefore, this algorithm uses two phases:

- exploration
- exploitation

The exploration phase is used to avoid local solutions according to extensive coverage. The exploitation is mainly used for accuracy enhancement for obtained solutions in the exploration phase [42].

3.1. The exploration phase

The AOA has used the simplest rule to simulate the behavior of AOA operators. The equations for position updating are as follows for the exploration section:

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} best(x_j) \div (MOP + \varepsilon) \times \left((UB_j - LB_j) * \mu + LB_j\right) & r2 < 0.5 \\ best(x_j) \times MOP \times ((UB_j - LB_j) * \mu + LB_j) & otherwise \end{cases} \quad (1)$$

Where $x_{i,j}^{(C_{Iter}+1)}$ denotes the i_{th} solution in the next iteration, $x_{i,j}^{(C_{Iter})}$denotes the j_{th} position of the i^{th} solution at the current iteration, and $best(x_j)$ is the j^{th} position in the best-obtained solution so far. ϵ is a small integer number, UB_j, and LB_jdenote to the upper bound value and lower bound value of the j^{th} position, respectively. μ is a control parameter to adjust the search process, fixed to 0.5[22].

3.2. The exploitation phase

In AOA, the exploitation operators of AOA explore the search area deeply on several dense regions and approach to find the best solution based on two search methods which are modeled as follows:

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} best(x_j) - MOP \times \left((UB_j - LB_j) * \mu + LB_j\right) & r3 < 0.5 \\ best(x_j) + MOP \times ((UB_j - LB_j) * \mu + LB_j) & otherwise \end{cases} \quad (2)$$

4. The proposed binary AOA

In this section, a new binary version of the arithmetic algorithm is introduced, considering that in this research, the offloading problem has been solved in the form of a binary problem. The exploration and exploitation mechanisms are achieved by the arithmetic operators' conception in math are presented by logical operators in the binary AOA. Before starting the AOA, the search phase (exploration or exploitation) should be selected. So a Math Optimizer Accelerated (MOA) function is a coefficient calculated by:

$$MOA(C\_Iter) = Min + C\_Iter(\frac{Max-Min}{M-Iter}) \quad (3)$$

Where *MOA (C_Iter)* denotes the function value at the *t*th iteration, *C_Iter*denotes the current iteration, which is between 1 and the maximum number of iterations (*M_Iter*), *Min* and *Max* denote the minimum and maximum values of the accelerated function, respectively. The flowchart of the proposed BAOA algorithm is shown in Figure 2.

4.1. The exploration phase

In this section, the exploratory phase of the BAOA is presented. According to the role of mathematical calculations using either XOR logical operator (Division) operator or even XOR logical operator (AND) and operator in high distributed values or decisions the arithmetic operators, the exploration search mechanism has been committed. This issue must be considered; the target could not be easily approached by the operators (D and M) due to their high dispersion, unlike other operators (S and A)[22].So the exploration operators (D and M) were applied to support the other stage (exploitation) through enhanced communication between them. To go to the exploration phase, a variable Math Optimizer Probability (MOP) is needed, which is a coefficient is obtained by:

$$MOP(C\_Iter) = 1 - \frac{C\_Iter^{1/\propto}}{M\_Iter^{1/\propto}} \quad (4)$$

Where MOP (C_Iter) denotes the function value at the tth iteration, C_Iter denotes the current iteration, and (M_Iter) denotes the maximum number of iterations. The $\propto$ is a sensitive parameter and defines the exploitation accuracy over the iterations, which is fixed equational to 5 according to the experiments. In this study, the sigmoid function is used to convert the MOP function to a binary function. Given that the range of sigmoid functions includes all real numbers and the return value of this function also changes uniformly from 0 to 1 or from 1 to -1 depending on the type of function.

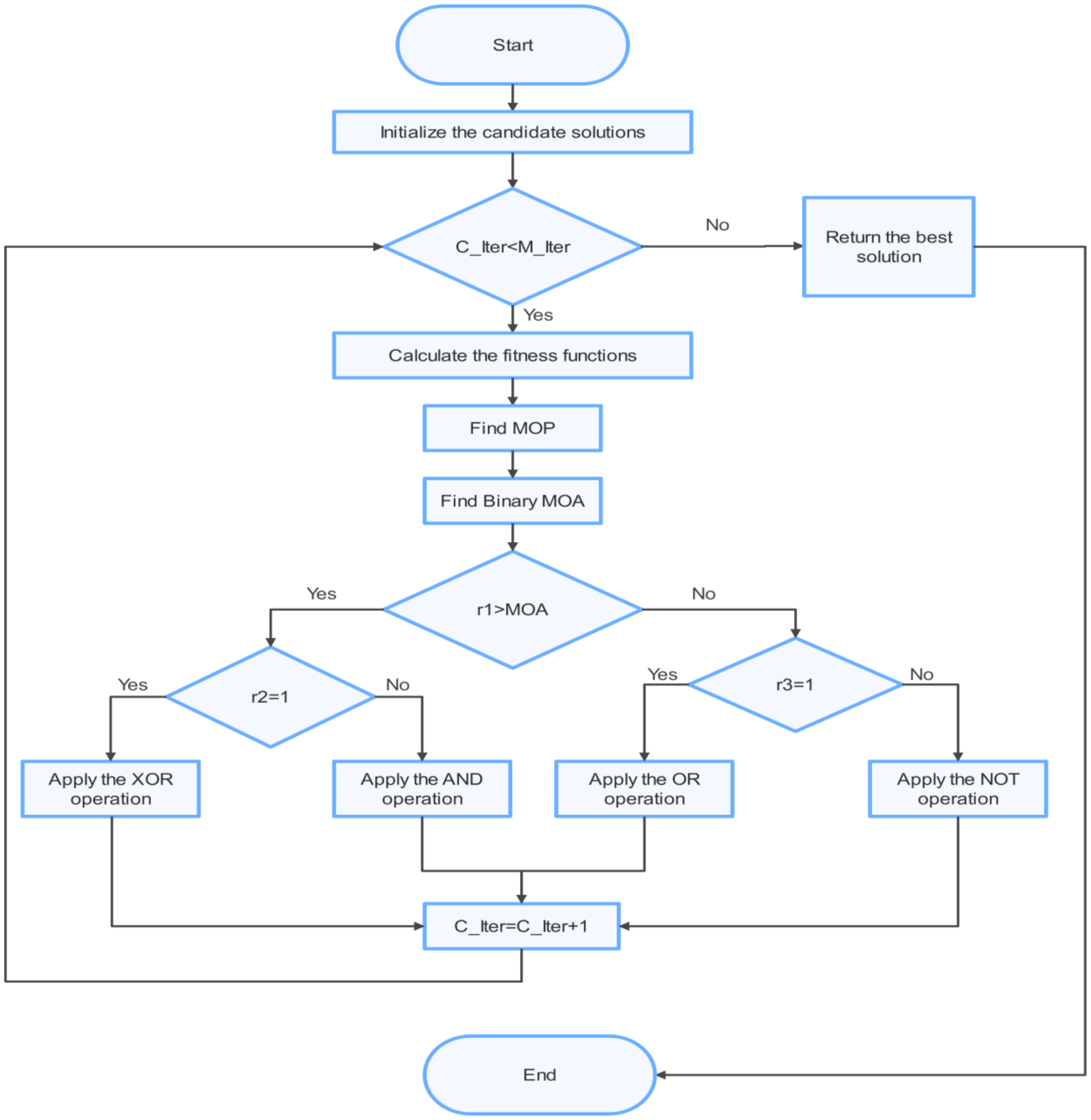


Figure 1. Flowchart of the proposed BAOA

The pseudo-code of the proposed BAOA algorithm is presented as follows. To convert a continuous MOP variable to the binary variable, the following formulas are used:

$$S\,(MOP(C\_Iter)) = \frac{1}{1-e^{-\mathrm{MOP(C\_Iter)}}} \tag{5}$$

The variable r2 is obtained to better examine the problem space, which is either 1 or 0, which is calculated by:

$$IF = \begin{cases} \theta < S\left(MOP(C_{Iter})\right) & r2 = 1 \\ Else & r2 = 0 \end{cases} \tag{6}$$

Where θ is a random variable. To achieve the binary exploration phase, the logical operators "Xor" and "And" are used instead of division and multiplication operators, respectively. So the search area is randomly explored to find the best solution based on "Xor" and "And" logical operators, which are presented in Equation 7.

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} best(x_j)\ xor\ (MOP+\varepsilon) & r2 = 1 \\ best(x_j)\ and\ MOP & r2 = 0 \end{cases} \quad (7)$$

Where $x_{i,j}^{(C_{Iter}+1)}$ denotes the ith solution in the next iteration, and best (x_j) is the position in the best-obtained solution so far. ϵ is a small integer number.

4.2. The exploitation phase

In this section, the Exploitation phase of binary BAOA is presented; either Subtraction (S) or Addition (A) got high-dense results in mathematical calculations. Because of the low dispersion of the operators (S and A), the target could be easily approached. The variable r3 is obtained to better examine the problem space, which is either 1 or 0, which is calculated by:

$$IF = \begin{cases} \theta < S\left(MOP(C_{Iter})\right) & r3 = 1 \\ Else & r3 = 0 \end{cases} \quad (8)$$

Where θ is a random variable. To achieve the binary exploitation phase, the logical operators OR and Not are used instead of Subtraction (S) or Addition (A) operators, respectively. So the search area is deeply explored to find the best solution based on "XOR" and "And" logical operators, which are modeled in Equation 9.

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} best(x_j) Not\ MOP & r3 = 1 \\ best(x_j)\ Or\ MOP & r3 = 0 \end{cases} \quad (9)$$

This process, besides exploration search strategies, assists to find the optimal solution so that the variety of problem solutions is satisfied. Algorithm 1 indicates the pseudo-code of the proposed BAOA algorithm.

5. Binary quantum arithmetic optimization algorithm

Quantum computing is a new theory that is the result of computer science and quantum mechanics. The primary purpose of quantum computing is to examine all possible solutions that the computer can access if provided by the laws of quantum mechanics. Over the past decade, quantum computing has received more attention than classical computing and has shown that it can be used as an efficient tool in problem-solving. It should be noted that if quantum mechanics is considered, the trajectory sentence is meaningless. Because $X_{i,d}^{t}$, and $V_{i,d}^{t}$ of a particle cannot be determined simultaneously according to the uncertainty principle[44].

Therefore, if the particles in a system behave quantum, the efficiency will be far removed from the classical AOA[22].Clerk and Kennedy have proposed a trajectory analysis in research[45], according to which it can be said that convergence in algorithms such as PSO occurs when each particle converges to its local adsorbent. Local charms are displayed with p_i, so p_i is:

$$p_i = (p_i.p_1.p_2.\dots.p_i.D) \quad (10)$$

And φ_d^t is calculated by:

$$\varphi_d^t = C_1 r_{i.d}^t / (C_1 r_{i.d}^t + C_2 R_{i.d}^t) \quad (11)$$

Where C1 and C2 are two constant acceleration coefficients and $r_{i,d}^t$ and $R_{i,d}^t$ are two random numbers with normal distributions in the range of 0 and 1. So every $p_i$is calculated as follows:

$$p_i = \varphi_d^t * Pbest_{i.d}^t + (1-\varphi_d^t) * x_*^t \quad (12)$$

Algorithm 1:Pseudo-code of the proposed BAOA algorithm

```
Initialize the Binary Arithmetic Optimization Algorithm parameters
2:    Initialize the population randomly.
3:    while (C_Iter < M_Iter) do
4:        Find the Fitness Function (F F) for solutions
5:        Find the best solution.
6:        Find the MOA parameter using Equation. (3).
7:        Find the MOP value using Equation. (4).
8:        for (i=1 to Solutions) do
9:            for ( j=1 to Positions) do
10:               Find random numbers[0, 1] (r1, r2, and r3)
11:               if r1 >MOA then
12:                   Exploration phase
13:                   if r2 =1 then
14:                       (1) Apply the Xor logical operator (XOR)
15:                       Update the i_th solutions' positions
16:                   else
17:                       (2) Apply the AND  logical operator (AND)
18:                       Update the i_th solutions' positions
19:                   end if
20:               else
21:                   Exploitation phase
22:                   if r3 =1 then
23:                       (1) Apply the OR math operator (OR).
24:                       Update the i_th solutions' positions
25:                   else
26:                       (2) Apply the NOT math operator (NOT).
27:                       Update the i_th solutions' positions
28:                   end if
29:               end if
30:           end for
31:       end for
32:       C_Iter=C_Iter+1
33:   end while
34: Return the best solution (x).
```

Where $Pbest_{i,d}^{t}$ refers to the best position the candidate i has ever had, and x_{*}^{t} refers to the global best candidate until itis repeated. Also, every single candidate in BQAOA is treated as a spin-less one moving in quantum space, and the probability of the candidates appearing at the position in the search iteration is determined from a probability density function. Employing the Monte Carlo method, each candidate behaviors with the following rules:

$$mbest_{d}^{t} = \frac{1}{N}\sum_{i=1}^{N} pbest_{i.d}^{t} \tag{13}$$

$$\alpha = \alpha_1 \frac{(T-t)*(\alpha_0-\alpha_1)}{T} \tag{14}$$

Where α_0 and α_1 are the initial and final values α, respectively; t is the maximum number of iterations; t is the current search iteration number.

If randv>= 0.5

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t + \alpha\left|\left(best(x_j)\ Xor\ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r2 = 1 \\ x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t + \alpha\left|\left(best(x_j)\ And\ \ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r2 = 0 \end{cases} \tag{15}$$

Else randv< 0.5

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t - \alpha\left|\left(best(x_j)\ Xor\ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r2 = 1 \\ x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t - \alpha\left|\left(best(x_j)\ And\ \ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r2 = 0 \end{cases}$$

If randv>= 0.5

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t + \alpha\left|\left(best(x_j)\ Not\ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r3 = 1 \\ x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t + \alpha\left|\left(best(x_j)\ Or\ \ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r3 = 0 \end{cases}$$

Else randv< 0.5 (16)

$$x_{i,j}^{(C_{Iter}+1)} = \begin{cases} x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t - \alpha\left|\left(best(x_j)\ Not\ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r3 = 1 \\ x_{i,j}^{(C_{Iter}+1)} = \varphi_d^t - \alpha\left|\left(best(x_j)\ Or\ \ MOP\right) - mbest\right| ln\left(\frac{1}{u_{i.d}^t}\right) & r3 = 0 \end{cases}$$

To provide the quantum version of the AOA, Equation 7 with Equation 15 and Equation 9 with Equation 16 are exchanged.

6. Proposed offloading scheme

The main steps of the proposed offloading schemes are shown in Figure 2. As shown in this figure, first a binary and quantum version AOA algorithm denoted as BQAOA are presented. In the second step, task offloading problem is formulated in the considered 3-tier architecture, in which various factors and constraints are used in this process. In the third step, a Markov model is presented for network bandwidth between IoT and MEC, which helps us in computing the average bandwidth between IoT and MEC. Then

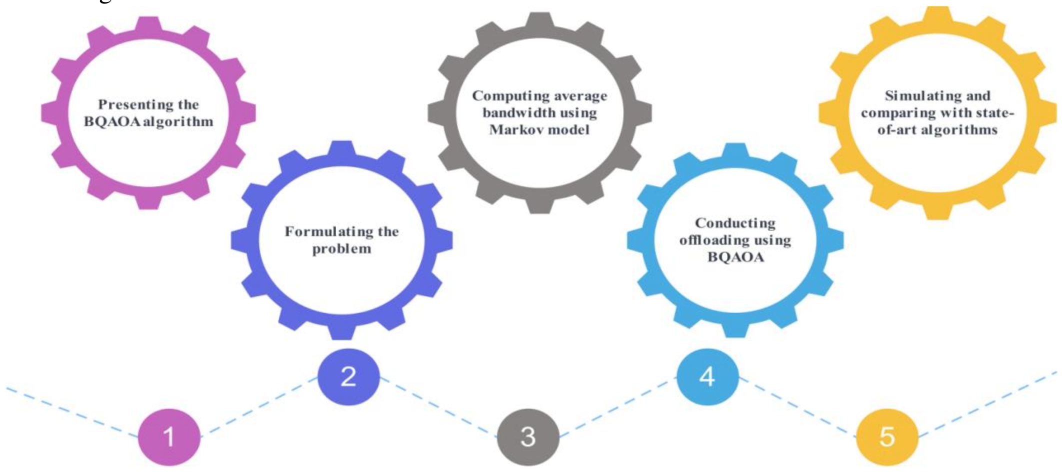

Presenting the BQAOA algorithm
Formulating the problem
Computing average bandwidth using Markov model
Conducting offloading using BQAOA
Simulating and comparing with state-of-art algorithms
1
2
3
4
5

Figure 2. The steps of BQAOA

6.1. Architecture

This research assumes that the computational application is processed first in the MDs themselves. If their computational requirements are not met in the MDs, the application is offloaded to the nearest computational server to small cell base stations (SBSs) antennas. Assuming that the computational power of sMEC, the closest server to SBSs, is not sufficient to process computational tasks, the processing power of mMEC, the nearest server to macro-base stations (MBSs), is used, which is at the edge of cloud computing and has more computational power. In this study, the advantages of MEC as architecture of edge computing are used to solve the lack of computational resources in MDs. In this way, the maximum processing resources available in both MDs and the nearest servers of MDs are used to reduce costs and time. Otherwise, the nearest processing server in edge computing is used using offloading solutions. Indeed for the lack of processing resources on MDs, the offloading solution is compensated[27, 46, 47]. So that once the processing tasks are offloaded to the nearest processing sMEC server and then to the processing servers in the nearest fog computing mMEC server. The main goal of these approaches is to perform a placement in a way that usage of computational resources in edge computing in the hybrid computing resources is convenient, the waste of them can be decreased, and the high QoS to meet the user's computational requirements is satisfied[48-51]. The outline of the conceptual model is shown in Figure 3.

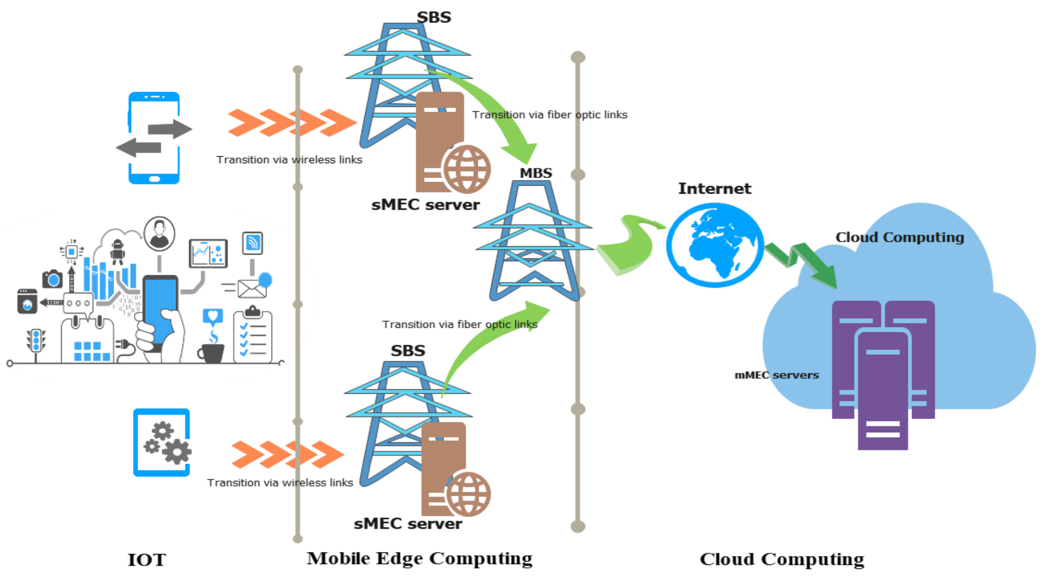

SBS
Transition via fiber optic links
Transition via wireless links
MBS
sMEC server
Internet
Cloud Computing
Transition via fiber optic links
SBS
mMEC servers
Transition via wireless links
sMEC server
IOT
Mobile Edge Computing
Cloud Computing

Figure 3. Task offloading in MEC-based 3-tier architecture

6.2. The problem formulation

In this study, multi-access edge computing in the 5G Heterogeneous network is assumed that the set of SBSs is connected to MBSs via optical fiber links, and MDs are connected to SBS via wireless links. According to the importance of reducing energy consumption on the part of MDs and increasing users' side QoS, two types of implementation models can be assumed to implement computational tasks.

- **User-side processing**: If the user's processing resources were sufficient to perform the computational tasks, the computational tasks would be executed on the MDs.
- **Offloading schemes**: If the computational tasks with high computational resource requirements that could not be satisfied by MDs processing capability:
    - **Offloading to SBS:** If the user's processing resources were not sufficient to perform the computational tasks, the computational tasks would be executed on the mMEC connected two SBSs via wireless links.
    - **Offloading to MBS:** If the computing resources on the SBS side were limited, the computing tasks would be offloaded to MBS connected to SBS via optical fiber links.

An integrated set of SBSs, MBSs, and MEC servers are assumed that computational Tasks could be served. SBSs and SMEs, MBS and mMEC are co-located, respectively. The connection between MDs and SBSs leads the reducing latency and increasing the transfer rate. So this assumption that the MDs were connected directly to the MBS is not considered. Because the MEC server has a deficiency of computing potency, large task offloading requests could make the SBS overloaded; in this case, the mMEC server will be another choice for available computational resources

[29]. The MDs are a set of M = {1,2,… M} and SBS as a set of N = {1,2,...N}. Each MD has one computing task to execute that can be executed on MD, sMEC, or mMEC, respectively. The computational task of MD can be offloaded to sMEC or be redirected to mMEC. The MD's computational tasks are indivisible and automated. The computational task is displayed as follows.

$$T_m = \{C_m . D_m\}. m \epsilon M \tag{19}$$

Where C_m, D_m are the total computations by the CPU cycle and the size of a computational task data, respectively.

6.2.1. Local execution model

In general, the energy consumption of smart devices is considered in four aspects, including sensing, actuation, processing, communication, and standby mode. Therefore, the energy consumption of each device is calculated using the following formula.

$$\mathrm{P_d(t) \triangleq S_d(t) + T_d(t) + C_d(t) + P_{std}} \tag{23}$$

$\mathrm{P_{std}}$is the energy required to stay in Standby mode, and $\mathrm{S_d(t)}$ is the energy required to receive and store data, $\mathrm{T_d(t)}$ is the energy required to transmit data at time t, and $\mathrm{C_d(t)}$ is the energy used to process the data. By integrating the above relation, the total energy consumed during the operation of the device is calculated through the following two formulas.

$$\mathrm{E_{S_d} \triangleq \int_{t_{S_d}}^{N} S_d(t)dt\ , E_{T_d} \triangleq \int_{t_{T_d}}^{N} T_d(t)dt,\ E_{C_d} \triangleq \int_{t_{C_d}}^{N} C_d(t)dt} \tag{24}$$

$$\mathrm{E_d(t) \triangleq E_{S_d} + E_{T_d} + E_{C_d} + E_{std}} \tag{25}$$

$$\mathrm{E_{d_{Total}} \triangleq \sum_{j}^{N} E_{S_{d_j}} + \sum_{k}^{N} E_{T_{d_K}} + \sum_{j}^{N} E_{C_{d_m}} + E_{std}} \tag{26}$$

$\mathrm{E_{d_{Total}}}$is the total energy consumption in each device and j = {1,…, N} and K={1,..,N} and M={1,…,N} and t is the operating time of each device. To meet the QoS, energy consumption and completion time parameters are considered. The local completion time for executing task T_m is calculated as follows.

$$T_m^l = \frac{C_m}{f_m^l} \tag{27}$$

WhereC_m,f_m^l are the total numbers of computational by the CPU cycle and local computability of MD_m, respectively. The E_m^l is the energy consumption of MD_m Task T_m is performed locally.

$$E_m^l = \propto C_m (f_m^l)^2 \tag{28}$$

The $\propto$ is the coefficient associated with the chip considered for energy consumption.

6.2.2. Computation-offloading Model

The offloading strategy is categorized into two schemes for offloading computational tasks via sMEC or via mMEC servers. This section discusses the total power consumption to complete the computational tasks, including offloading for the task execution on the MEC servers and returning the result data. The improvement is considered from the user's vision, so, except for the amount of performing offloading, the energy consumption of MEC servers is not considered[52].

6.2.2.1. sMEC Offloading

The completion time T_{mn}^{sbs} for execution task,I_m on sMEC servers via wireless links is the sum of transfer time T_{mn}^{t} of task I_m to sMEC and remote computation time T_{mn}^{es}. The wireless transfer time when MD_m transmits its tasks to sMEC is calculated as follows.

$$T_{mn}^t = \frac{D_m}{R_{mn}} \tag{29}$$

Where $\mathrm{D_m}$, $\mathrm{R_{mn}}$ is the size of a computational task data and the communication model, respectively. The computational time required to execute computational Tasks on sMEC n is given by:

$$T_{mn}^{es} = \frac{C_m}{f_{mn}} \tag{30}$$

Where C_m , f_{mn} are the total computational numbersby the CPU cycle and the computational resource allocated to MD$_m$ via sMEC. According to formulas 21 and 22, the completion time of T_{mn}^{sbs} can be calculated as follows:

$$T_{mn}^{sbs} = T_{mn}^{t} + T_{mn}^{es} = \frac{D_m}{R_{mn}} + \frac{C_m}{f_{mn}} \tag{31}$$

As mentioned, the power consumption required to load the Computational Task I$_m$ to run in sMEC$_n$ is E_{mn}^{t}Where is the power consumption required to transfer the Task I$_m$ from MD$_m$ to SBS$_n$.

$$E_{mn}^{sbs} = E_{mn}^{t} = p_m T_{mn}^{t} = p_m \frac{D_m}{R_{mn}} \tag{32}$$

Where p_m is constant transmission power and T_{mn}^{t} is calculated in formula 17

Table 2. Notations

Item	**Description**
T_m	The computational task
D_m	The size of a computational task data
C_m	The total number of computational by the CPU cycle
w	Channel bandwidth
p_m	Constant transmission power
h_{mn}	Channel gain between MDm and SBSm
n_0	The noise power
Q_{mn}	The inter-cell interface
f_m^l	Local computability of MDm[34]
$\propto$	The coefficient associated with the chip
f_{mn}	The computational resource allocated to MD$_m$via sMEC n
f_0	The computational resource allocated by mMEC to each MD is considered to be fixed and unique for all MDFs
R_{mn}	The communication model
T_m^l	The local completion time
E_m^l	The energy consumption of MD$_m$
T_{mn}^t	The transfer time of task I$_m$ to sMEC
T_{mn}^{es}	The remote computation time T_{mn}^{es}
T_{mn}^{sbs}	The execution time for the task I$_m$ on sMEC servers
E_{mn}^{sbs}	The power consumption required to load the Computational Task I$_m$ to run in sMEC$_n$
E_{mn}^t	The power consumption required to transfer the Task Im from MDm to SBSn
T_{mn}^{tm}	The transmission time when I$_m$ computational tasks migrate from SBS$_n$ to mMEC servers
T_{mn}^{em}	The Computation time for Computational Task on mMEC server
T_{mn}^{mbs}	The completion time when the computational task m is executed on mMEC servers
E_{mn}^{mbs}	The power consumption required to load the Computational Task I$_m$ to run in sMEC$_n$

6.2.2.2. mMEC offloading

Computational tasks can be performed on the mMEC servers by migrating from the SBS connected to MBS via the fiber optic link. The T_{mn}^{tm}, the transmission time when I$_m$ computational tasks migrate from SBS$_n$ to mMEC servers is calculated by:

$$T_{mn}^{tm} = \frac{D_m}{\beta} \tag{33}$$

Where D_m, β are the size of a computational task data and the fiber link transfer rate between each SBS and MBS, respectively. The T_{mn}^{tm} is the computation time for Computational Task on mMEC server is given by:

$$T_{mn}^{em} = \frac{C_m}{f_0} \tag{34}$$

Where $f_0$is considered the computational resource allocated by mMEC to each MD intended for offloading, which is fixed and unique for all MDFs [52]. C_m is the total number of computational by the CPU cycle. When the computational task m is executed on mMEC servers, the completion time T_{mn}^{mbs} is the sum of the transmission time T_{mn}^{t} ,T_{mn}^{tm}, and T_{mn}^{em} as given by:

$$T_{mn}^{mbs} = T_{mn}^{t} + T_{mn}^{tm} + T_{mn}^{em} = \frac{D_m}{R_{mn}} + \frac{D_m}{\beta} + \frac{C_m}{f_0} \tag{35}$$

Similarly, the energy consumption of Task I_m offloaded to mMEC server is the energy consumption of E_{mn}^{t} and can be calculated as follows:

$$E_{mn}^{mbs} = E_{mn}^{t} = p_m \frac{D_m}{R_{mn}} \tag{36}$$

Where p_m is constant transmission power and E_{mn}^{t} is the power consumption required to transfer the task I_m from MD_m to $SBS_{n.}$

6.2.3. Communication model

In this section, the communication model is presented with $R_{mn}$for offloading tasks to sMEC or mMEC. The MD's computational task can be offloaded to the local MD and the MEC servers according to their computational capacity. The orthogonal frequency division multiple accesses for communication between MDs and SBSs is assumed in this study. The transfer rate of the computational task from MD_m to SBS_n is calculated as follows.

$$R_{mn} = w\log_2^{\left(1+\frac{p_m h_{mn}}{n_0+Q_{mn}}\right)} \tag{20}$$

Where p_m,h_{mn}, n_0, Q_{mn} , and *w are* constant transmission power, Channel gain between MD_m and SBS_m, noise power, inter-cell interface, and channel bandwidth, respectively. During time, network bandwidth between MDs and MEC servers varies. In this study, a Markov model is used to achieve the average amount of bandwidth. In the proposed Markov model, the sets S = {S_1, S_2, S_3} denote the existence states, in which each state denoted to the different level of bandwidth. Figure 4 depicts the transition diagram of the applied Markov model for network bandwidth Markov model to show the sequence of decision-making outcomes. It is also possible to navigate between states by a probability distribution represented by the form of the transition matrix with three transition states as unknown parameters. The probability matrix can be written as follows:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{21}$$

According to the above discussion, the Markov model could be defined as the model *A*=(*S, P*) and the sequence *S*= S_1, S_2, $S_3 \in S$ .In the Markov model, generally, this questions should be answered:

- ✓ Having the sequence *S* and the model *A*, what is the optimal sequence of $T = S_1, \ldots, S_3$? to answer this question, the QABOA algorithm is generally applied.

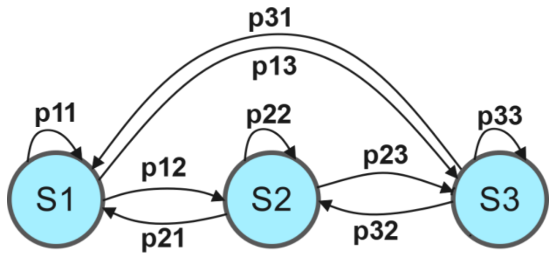


Figure 4: The transition diagram applied Markov model for network bandwidth

In this scheme, we compute the average available bandwidth between local MDs, sMEC, and mMEC server by using Equation 22.

$$\pi = \left(\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}\right)^n$$

$$\pi = (\pi_1, \pi_2, \pi_3) \tag{22}$$

$$Bandwidth = \sum_{k=1}^{3} Bandwidth_k * \pi_k \quad where \ \sum_{k=1}^{3} \pi_k = 1$$

In which, π is obtained by successive multiplication of matrix P, bandwidth$_k$ indicates the bandwidth of the k$_{th}$ state also, in this equation; $\pi_k$is the probability of the i$_{th}$ state of the proposed Markov model. By solving the energy consumption Markov model for each local MDs, sMEC, and mMEC server, we can compute the average available bandwidth for a cloud environment. The Markov model is applied to predict the best offloading path between the states according to the following state to achieve better resource management.

6.2.4. Fitness function

The offloading decision determines the execution location for computational tasks such as sMEC offloading or mMEC offloading. Therefore, the decision has two phases based on the computational offloading scheme. The first involves offloading tasks to the MEC server via SBS, and the second involves offloading tasks from sMEC or mMEC. Let consider γ as a decision-offloading matrix,

$$\gamma = \{y_{mn} | m \in M. n \in N\} \tag{37}$$

Each y_{mn} can be computed as follows:

$$y_{mn} = \begin{cases} 1. \ if \ the \ computational \ tasks \ offload \ to \ sMECn \\ 0. \qquad \qquad \qquad \qquad \qquad \qquad otherwise \end{cases} \tag{38}$$

Because the task is offloaded to at least one MEC server, this condition must be met.

$$\sum_{n=1}^{N} y_{mn} \le 1. \ \forall m \in M \tag{39}$$

Let consider z an offloading-location matrix for task I$_m$:

$$z = \{z_{mn} | m \in M. n \in N\} \tag{40}$$

And each element z_{mn} is calculated by:

$$z_{mn} = \begin{cases} 1. If \ the \ task \ I_m \ is \ executed \ on \ sMEC \ n \\ 0. \ if \ the \ task \ I_m \ is \ executed \ on \ mMEC \ n \end{cases} \tag{41}$$

The following condition ensures that each task must execute on at least one MEC server.

$$\sum_{n=1}^{N} z_{mn} \le 1. \ \forall m \in M \tag{42}$$

Consider M_{off} as a set of MDs which tasks are offloaded to sMECs and

$$M_{off} = \{m \in M. n \in M \ | y_{mn} = 1. z_{mn} = 1\} \tag{43}$$

And M_n is considered a set of MDs which tasks are offloaded to sMECm.

$$M_n = \{m \in M \ | y_{mn} = 1. z_{mn} = 1\} \tag{44}$$

Finally, we consider the computational resource matrix F for sMEC.

$$F = \{f_{mn} | m \in M. n \in N\} \tag{45}$$

Where $f_{mn} > 0$ is the resource allocation of sMECs to computational taskI_m. Because of the lack of computational resources, the following condition must be met.

$$\sum m \in M_{n.} f_{mn} \le f_n^{max}. \forall n \in N \tag{46}$$

Where f_n^{max} is the maximum computational capability of sMEC$_n$. This condition shows sMECm does not allocate any computational resource to MD$_m$.

$$f_{mn} = 0 \; if \; m \in M_n \tag{47}$$

Total completion time for execution task I$_m$ is given by:

$$T_m = \sum_{n=1}^{N} y_{mn}\left(T_{mn}^{t} + z_{mn}T_{mn}^{es} + (1-z_{mn})(T_{mn}^{tm} + T_{mn}^{em})\right) + \left(1-\sum\nolimits_{n=1}^{N} y_{mn}\right)T_m^{l} \tag{48}$$

The total energy consumption required to complete Task I$_m$ is calculated by

$$E_m = \sum_{n=1}^{N} y_{mn}\left(z_{mn}E_{mn}^{sbs} + (1-z_{mn})E_{mn}^{mbs}\right) + \left(1-\sum\nolimits_{n=1}^{N} y_{mn}\right)E_m^{l} \tag{49}$$

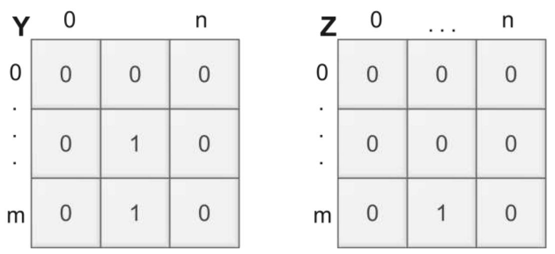
Therefore, the optimization problem can be shown as follows:

$$\begin{gathered} minimize\ (Y, F, Z) \sum_{m=1}^{M} (\eta_m^e E_m + \eta_m^t T_m) \\ \text{C1: } y_{mn}.\, z_{mn} \in \{0,1\}.\, \forall m \in M.\, \forall n \in N \\ \text{C2: } \textstyle\sum_{n=1}^{N} y_{mn} \le 1.\ \forall m \in M \\ \text{C3: } \textstyle\sum_{n=1}^{N} z_{mn} \le 1.\ \forall m \in M \\ \text{C4: } \textstyle\sum_{n=1}^{N} y_{mn} \le H.\ \forall n \in N \\ \text{C5: } f_{mn} > 0.\, \forall m \in M_{off}.\, \forall n \in N \\ \text{C6: } \textstyle\sum_{m \in M_n} f_{mn} \le f_n^{max}\ .\, \forall n \in N \end{gathered} \tag{50}$$

Where $\eta_m^e + \eta_m^t = 1$ and $\eta_m^e . \eta_m^t \in [0,1]$. Indicates weight by the energy consumption the completion time of MD$_m$ and H represents the number of sub-carriers. The C1 constraint is a binary offloading decision, and the C$_2$ and C$_3$ constraints ensure that each MD$_m$ can select at least one SBS. Constraint 4 indicates that the number of Tasks uploaded to the MEC server must be less than the total number of subcarriers. Constraints 5 and 6 indicate that the total computational resources allocated to MDF offloading by each sMEC cannot exceed its computational capacity. Formula 40 and 41 is an MINLP problem of NP-hard complexity and cannot be solved in polynomial time [52]. However, global improvement can be used to find the optimal solution to this problem. Worst computational complexity limits the exponential nature of its applications to 5G wireless networks, leading to high levels of communication, heterogeneous QoS requirements, and highly dynamic wireless channels. In this study, an efficient, low-complexity algorithm can obtain a high-quality solution, and convergence guaranteed using the AOA is developed.

6.3. The solution of BQAOA

In this study, each solution is implemented by a three-dimensional matrix with a size of $y(2, m, n)$, where m indicates the number of MDs and n indicates the number of sMECs. The values of $y(0, m, n)$ can be zero or one. The number one means that information is offloaded to process on sMEC and the number zero means that tasks are processed locally on MDs. Also, the values of $y(1, m, n)$ can be zero or one. The number one means that the tasks should be offloaded on mMEC, and the number zero means tasks should be offloaded on sMEC. In this study, instead of $y(0, m, n)$ and $y(1, m, n)$, $y(m, n)$ and $z(m, n)$ are used respectively for easier understanding.



Y	0		n
0	0	0	0
. . .	0	1	0
m	0	1	0

Z	0	. . .	n
0	0	0	0
. . .	0	0	0
m	0	1	0

Figure 5. The encoding of BQAOA's solutions

Figure 5 shows the encoding of each candidate solution. The number of both matrices' columns, Y and Z, is equal to N. The number of sMECs and m indicates the number of rows for matrices "Y and Z" and the number of MDs, respectively. For row 0in the matrix Y, all columns are equal to zero, which means that tasks on mobile device 0 must be processed locally and in the mobile device 0 itself without offloading to any sMEC server. The Y[1,1]=1 and Z[1,1]=0 show that the task on mobile device 1 must be offloaded on sMEC 1 and processed at the same sMEC server. The following example, Y[2,1]=1, shows that the task on mobile device 2 must be offloaded to sMEC 1, and the Z[2,1]=1 shows that the task could not be processed on sMEC 1 and must be offloaded to mMEC server.

7. Performance evaluation

In this section, the performance of the BQAOA method is presented. The CloudSim5 simulator, a toolkit for modeling and simulating extensible clouds, evaluates the performance. As a completely customizable tool, cloudSim5 allows the extension and definition of policies in all the software stack components, making it a suitable research tool to handle the complexities arising from simulated environments[43]. In this study, two different scenarios are considered to simulate the 5G-Net environment. Scenario A is assumed to be small with 50 within a coverage area (250 × 250 m2) MDs, 6 SBS, and 1 MBs connected to all SBSs, but Scenario B considers the problem space to be larger and the number of MDs, SBS, and MBS are considered 200 within a coverage area (1000× 1000 m2), 12 and 1 connected to all SBSs, respectively. Other standard simulation parameters used in both scenarios are shown in Table 3.

7.1. Simulation results

The performance of BQAOA has been compared with AOA, ICA, GA, and PSO, and Krill Herd algorithms in terms of energy consumption, processing time, and ESV, respectively. The energy consumption in this study refers to the electricity consumed by MDs, sMEC servers, which are the servers that are located next to the SBS, and also the electricity consumed by sMEC servers, the central cloud computing server. The purpose is to optimize the total set of energy consumed in these layers. Processing time refers to the time required to process all tasks on all MDs, sMEC servers, and mMEC servers, which is called makespan time. The ESV metric is also a parameter obtained by multiplying makespan time in energy consumption, calculated in Equation 49.

$$ESV=(Energy*makespan)/1000 \tag{51}$$

There is a trade-off between energy consumption and processing time, that if servers are used too much, time decreases and energy increases, and if servers are used less, time increases and energy decreases. In order to be able to decide, despite this trade-off, whether reducing energy consumption harms processing time, the existence of the ESV parameter is essential.

Table 3. Simulation parameters in Scenario A and B

Parameter	Value
Channel-gain	The path-loss model according to 140.7 + 36.7log10(d). where *d* (km) is the distance between the MD and the SBS
The noise power	n_0= -100 dBm
The number of MDs in Scenario A	10-45
The number of MDs in Scenario B	20-200
The number of tasks in Scenario A	10-45
The number of tasks in Scenario B	20-200
The transmit power of MD	p_m= 23 dBm
The data rate of the fiber-optic link	β= 1 Gbps
The number of subcarriers	H = 12
The bandwidth of each sub-channel	W = 3 MHz
The input data size of the computational tasks	[600, 1200] KB
The number of CPU cycles	Cm = [500, 1000] Megacycles
The CPU speed of the mMEC	f_0= 3 GHz,
The maximum CPU computing capacity sMEC	f_{max}= 6 GHz
The local computing power of the MDs	{0.5, 0.8, 1}GHz , $a = 5 \times 10^{-27}$
The weighting parameters of the computational task	0.5
Maximum number of iterations (T_{max})	1000
The size of the particle population (K)	30

7.1.1. Analysis BQAOA in terms of energy consumption in scenario A

For the purpose of this study, energy consumption refers to the energy consumed by MDs and sMECs without considering mMEC. In Figure 6, the horizontal axis shows the number of MDs and the vertical axis shows the amount of energy consumption in watts per hour.

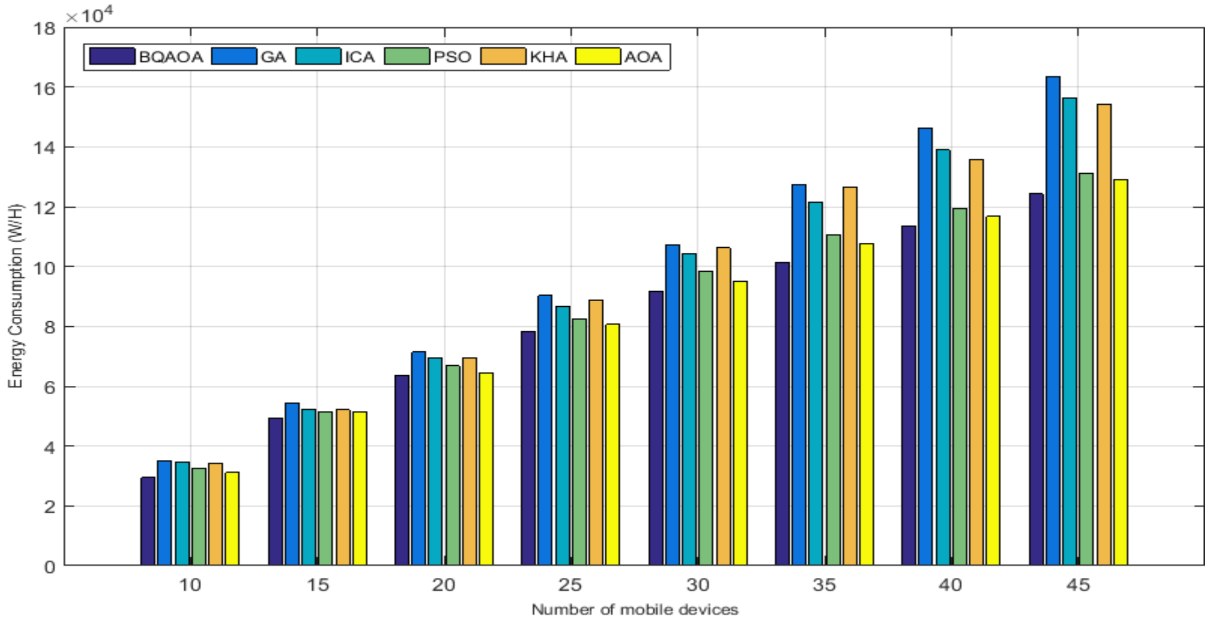


Figure 6. The Energy Consumption in Scenario A

The BQAOA algorithm has consumed less electricity than other algorithms in all cases and with any number of MDs while the genetic algorithm has consumed the highest amount of energy compared to other algorithms. As shown here, increased MDs was directly related to energy consumption which is due to an increase in the number of tasks as each mobile device has at least one task to perform.

According to Figure 7, the performance of BQAOA was better than AOA, GA, ICA,PSO and Krill Herd regarding energy consumption. Improvement in energy consumption compared to other methods is shown in Figure 7 in which the horizontal axis indicates the number of MDs and the vertical axis indicates the percentage of improvement in energy consumption.

The BQAOA algorithm showed the most significant improvement over the genetic algorithm. Furthermore, it is observed that with an increase in the number of MDs, improvement in the proposed algorithm becomes significantly better compared to other algorithms. This is due to the enlargement of problem space as, in larger problem spaces, the algorithms are optimally localized resulting in weaker performance. Still, the BQAOA algorithm was less involved in the local optimization.

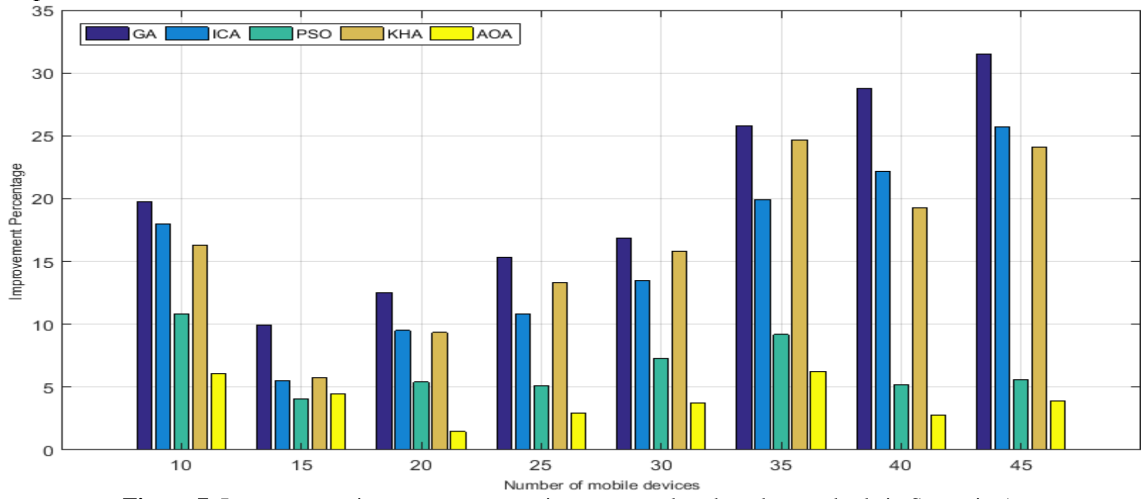


Figure 7. Improvement in energy consumption compared to the other methods in Scenario A

7.1.2. Analysis BQAOA in terms of energy consumption in scenario B

In Figure 8, the vertical axis shows the amount of energy consumption in watts per hour. In scenario B, the BQAOA algorithm consumed less energy than the other algorithms. Moreover, the genetic algorithm and the imperialism competition algorithm showed the highest electricity consumption, respectively. In addition, with an increase in the number of MDs, difference in algorithms' energy consumption becomes more significant which is due to the small size of the space problem when the number of MDs is low.

According to Figure 9, the performance of BQAOA was better than AOA, GA, ICA and PSO in terms of energy consumption. Figure 9 shows improvement in energy consumption in which the horizontal axis indicates the number of MDs and the vertical axis indicates the percentage of improvement in energy consumption compared to the other methods. Looking closely at Figure 9, the BQAOA algorithm showed the least improvement over the PSO and AOA algorithms because of the relatively good performance of these two algorithms compared to the other algorithms such as genetics, ICA, and KHA. In addition, the percentage of improvement will increase with an increase in the number of MDs. The difference in the performance of algorithms in a larger space could be seen better.

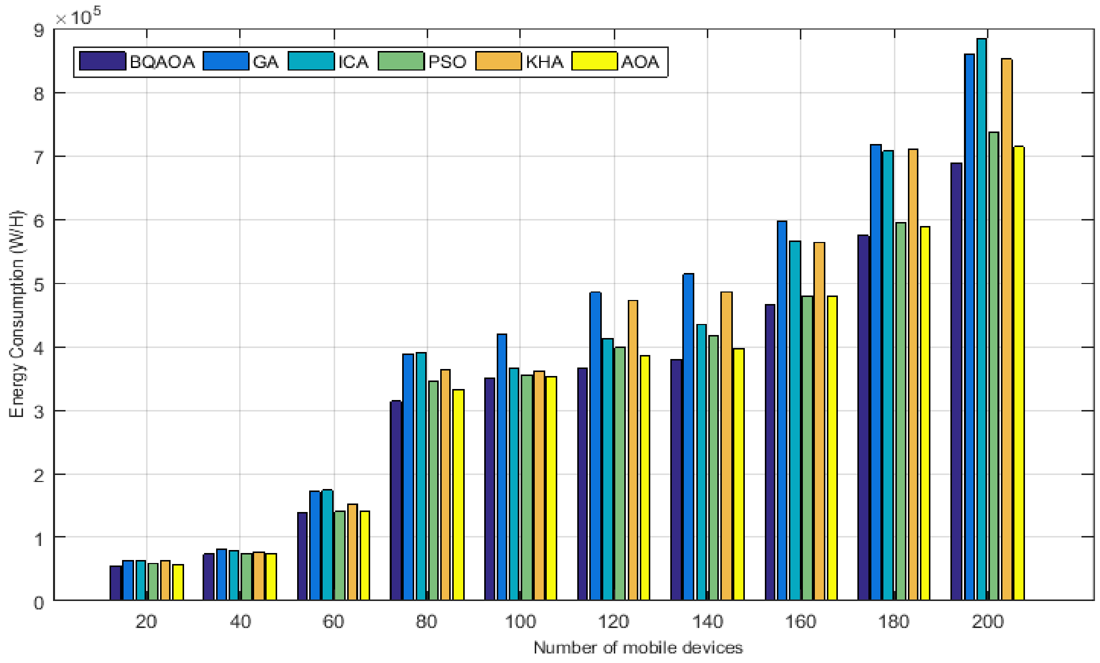


Figure 8. Energy consumption in Scenario B

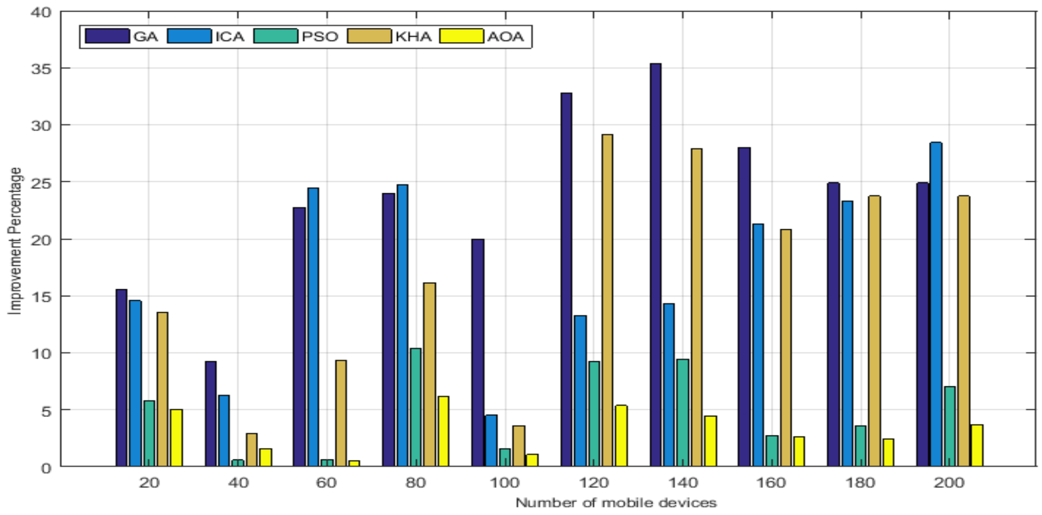


Figure 9. Improvement in energy consumption of BQAOA compared to the other methods in Scenario B

7.1.3. Analysis of BQAOA in terms of Makespan in scenario A

In this study, the makespan indicates full processing of all tasks on all MDs, whether locally or nor, on sMEC, or mMEC. Figure 10 shows the makespan time in which the horizontal axis indicates the number of MDs and the vertical axis indicates the makespan in milliseconds. Figure 11 shows an improvement in the processing time compared to the other methods where the horizontal axis indicates the number of MDs and the vertical graph indicates the percentage improvement in makespan.

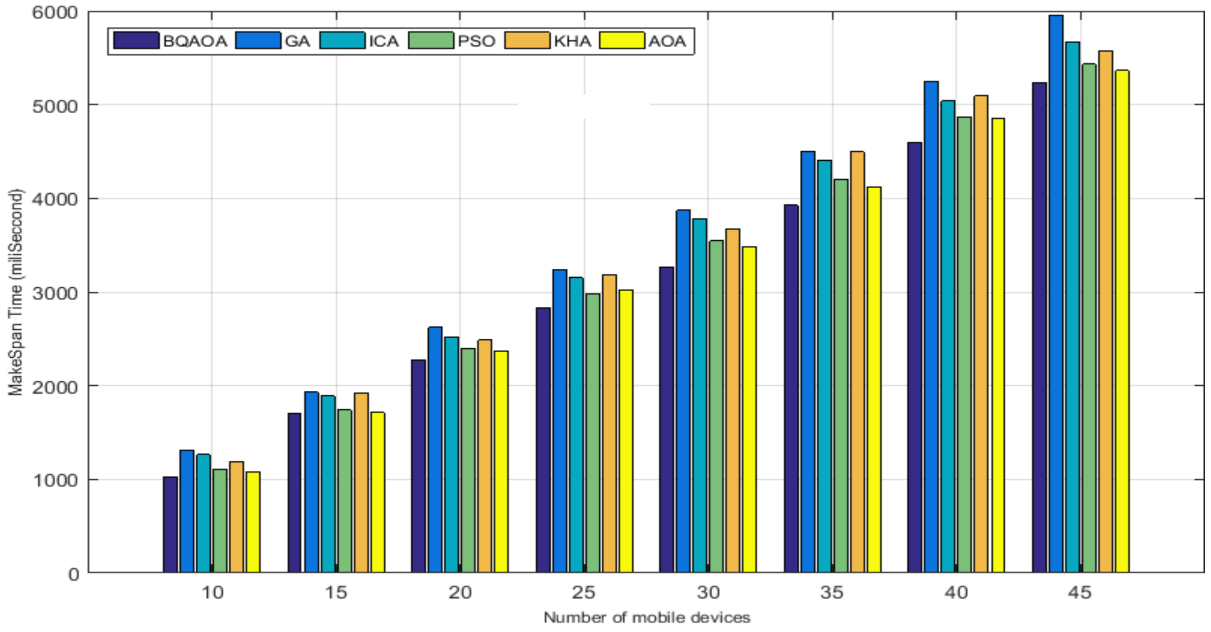


Figure 10. Makespan time in scenario A

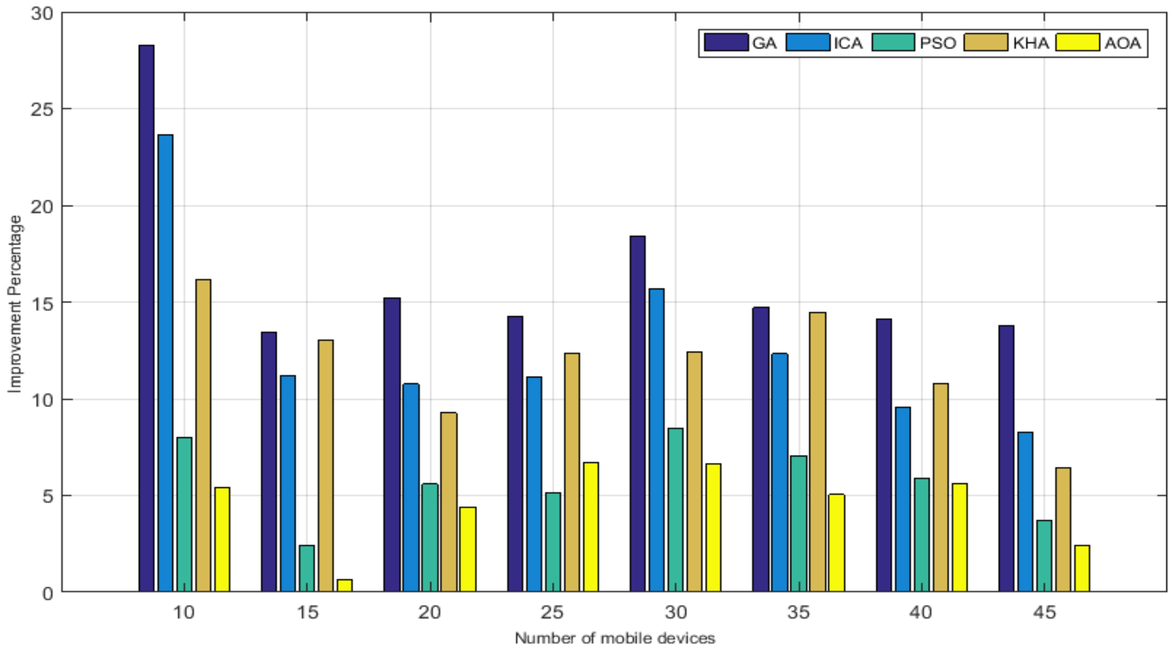


Figure 11. Improvement of makespan of BQAOA compared to the other methods in Scenario A

As shown in Figure 11, the BQAOA makespan algorithm was timeless compared to the other algorithms. The ICA algorithm had more makespan time so the BQAOA algorithm performed better than the other algorithms due to the reduced makespan time and the ICA algorithm had a relatively weak performance. The BQAOA algorithm has improved most in terms of makespan time compared to the genetic algorithm and ICA. This indicates the poor performance of these two algorithms in terms of makespan time.

7.1.4. Analysis BQAOA in terms of Makespan in Scenario B

In this section, the processing times is related to all tasks that are fully processed on all MDs, whether locally or not, on sMEC, or mMEC, . The horizontal axis indicates the number of MDs and the vertical axis shows the makespan time in milliseconds, as shown in Figure 12.

In Scenario B, the BQAOA algorithm achieved less makespan than the other algorithms, and the genetic algorithm and ICA performed worse than the other algorithms. Besides, the makespan time achieved by them was higher than the other algorithms. It was also observed that with an increase in the number of MDs, the makespan time increases which is due to is the fact that every mobile device has at least one task to perform, and therefore with the increase in MDs, the number of tasks increases and as a result, makespan time increases.

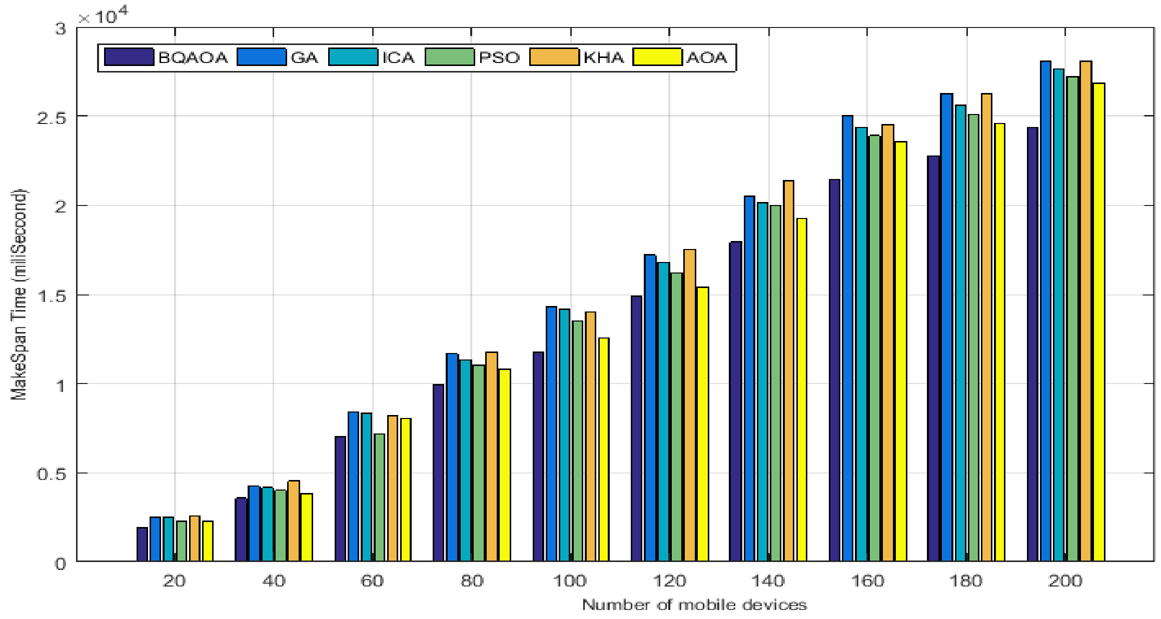


Figure 12.Makespan time in Scenario B

Figure 13 shows an improvement in the processing time compared to the other methods in scenario B. The horizontal axis refers to the number of MDs and the vertical axis refers to the percentage improvement in the processing time.

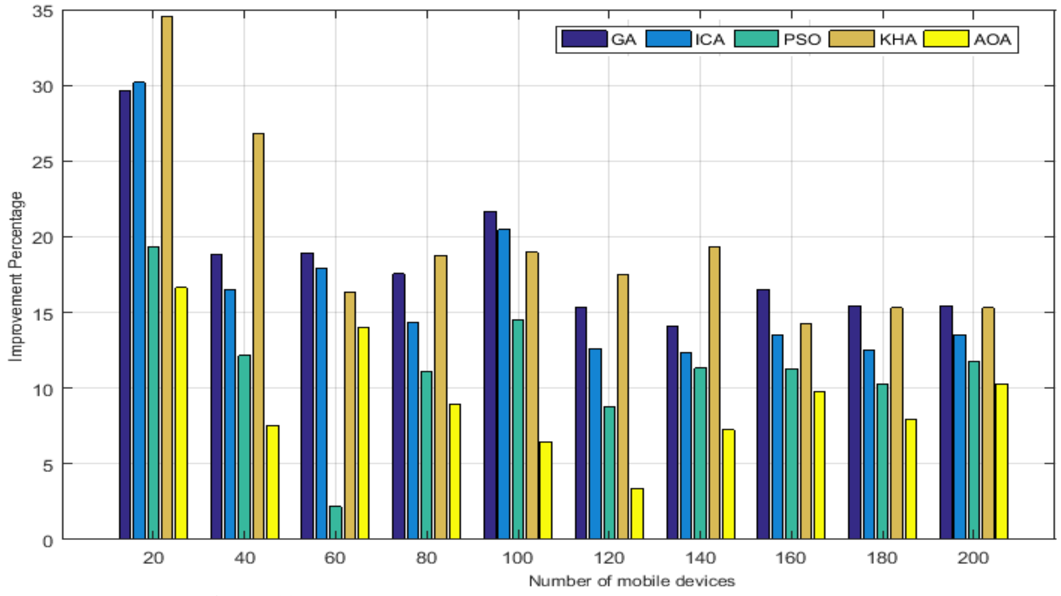


Figure 13. The makespan of BQAOA compared to the other methods in Scenario B

According to Figure 13, the lowest rate of improvement in the BQAOA algorithm compared to the other algorithms was related to the PSO algorithm. This indicated better performance of the PSO algorithm than the other algorithms after the BQAOA algorithm. Moreover, increased number of MDs improved the BQAOA algorithm more significantly than the other algorithms which was due to an increase in the problem space.

7.1.5. Analysis of BQAOA in terms of ESV in scenario A

Sometimes, reduced energy consumption may occur as a result of reduced number of processing elements, and in fact, this reduction in energy consumption increases the processing time. To measure BQAOA efficiency in reducing energy consumption, a metric called ESV was used which is calculated in Equation 49.
According to Figure 14, it can be argued that the performance of BQAOA in terms of ESV metric was better than AOA, genetic methods, ICA and PSO, and krill herd algorithms. Moreover, with an increase in the number of MDs, the BQAOA algorithm improves more than the other algorithms. Based on this, it is believed that the BQAOA algorithm affects improvement compared to the other algorithms in terms of both energy consumption and makespan time.

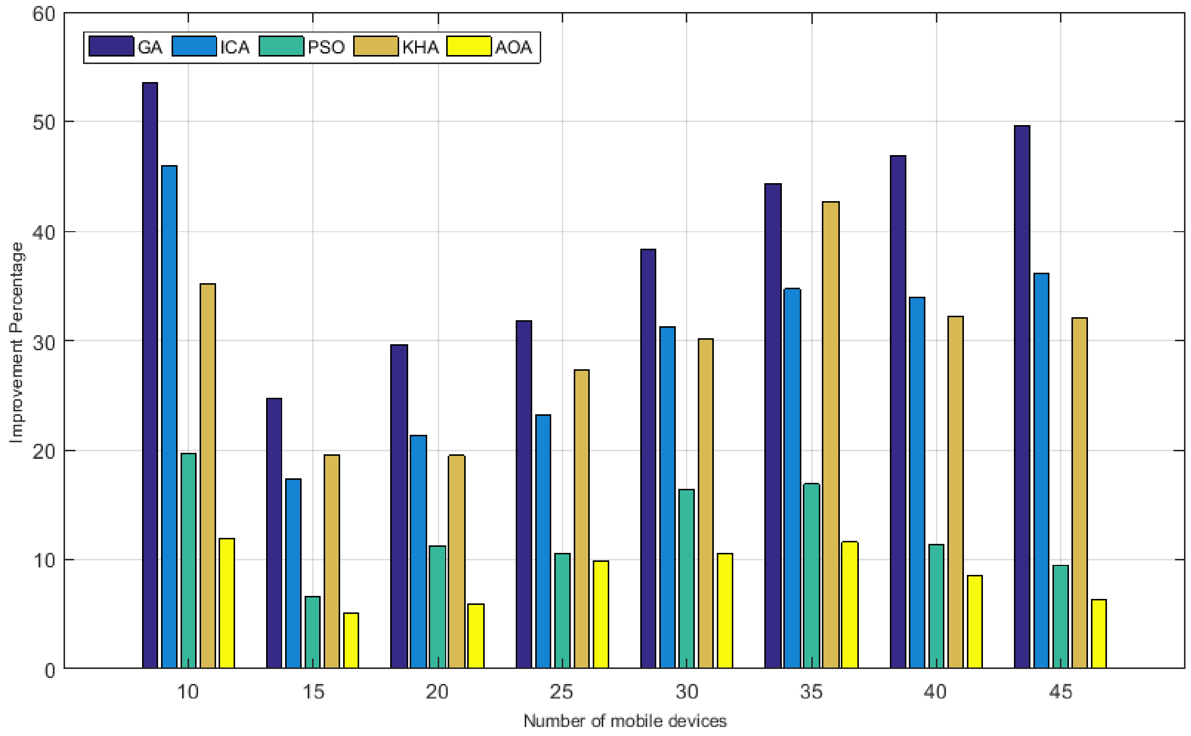


Figure 14. Improvement in the makespan of BQAOA compared to the other methods in Scenario A

Figure 14 shows the improvement of ESV compared to the other methods. The horizontal axis indicates the number of MDs, and the vertical axis shows the percentage of ESV recovery. According to Figure 14, the BQAOA algorithm had the best improvement in terms of ESV metrics compared to the genetic algorithm. Therefore, it can be concluded that the genetic algorithm generally showed less improvement than the other algorithms in terms of energy consumption and makespan time, and QBAO, AOA, and PSO algorithms had the best performances, respectively.

7.1.6. Analysis of BQAOA in terms of ESV in Scenario B

To measure the efficiency of BQAOA in reducing energy consumption, a metric called ESV was used according to Equation 51. Figure 15 shows performance of the BQAOA algorithm and the other algorithms in terms of ESV metrics in Scenario B. The horizontal axis represents the number of MDs and the vertical axis represents the metric value of ESV. The lower the ESV value, the better the performance of that algorithm in energy consumption and makespan time. According to Figure 15, it can be argued that the performance of BQAOA in terms of ESV metric was better than the AOA, genetic methods, ICA and PSO, and krill herd algorithms.

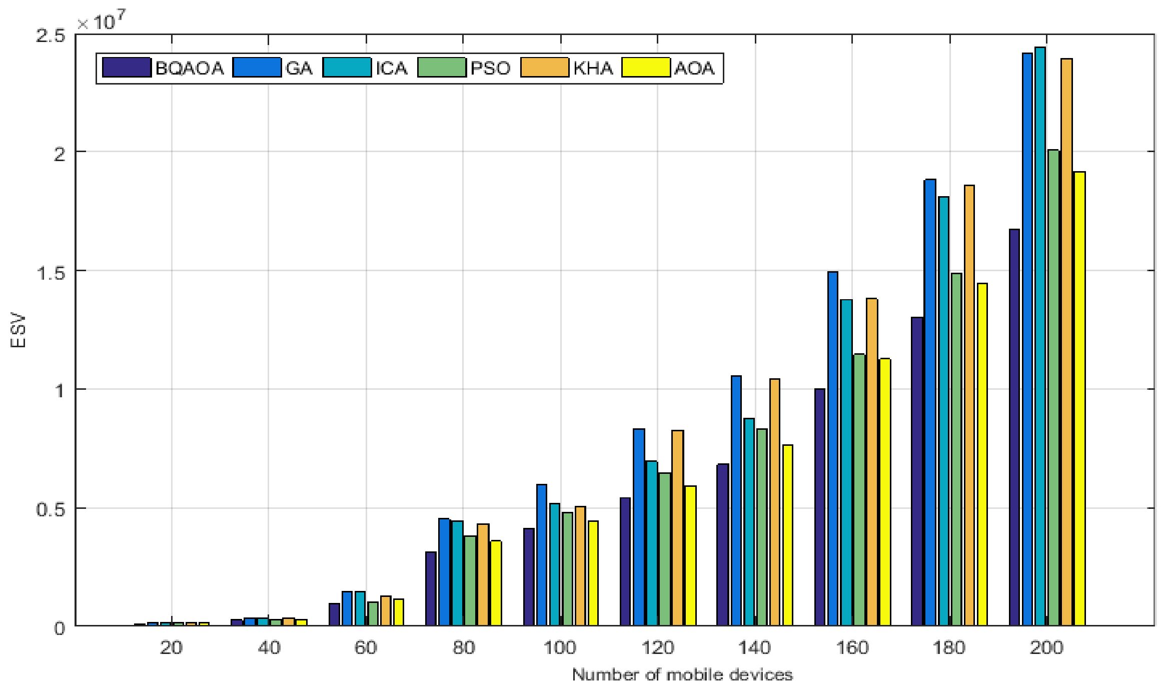


Figure 15. The ESV rate in Scenario B

Figure 16 shows the improvement of ESV compared to the other methods. The horizontal axis indicates the number of MDs and the vertical axis indicates the percentage of ESV recovery. Looking closely at Figure 16, it can be argued that the BQAOA algorithm showed the least improvement compared to the AOA and PSO algorithms which may be due to the relatively better performance of these two algorithms compared to the other algorithms and a higher rate of the BQAOA algorithm's improvement compared to the GA and KHA algorithms. The genetic algorithm and KHA performed worse in energy consumption and makespan time than the other algorithms in Scenario B.

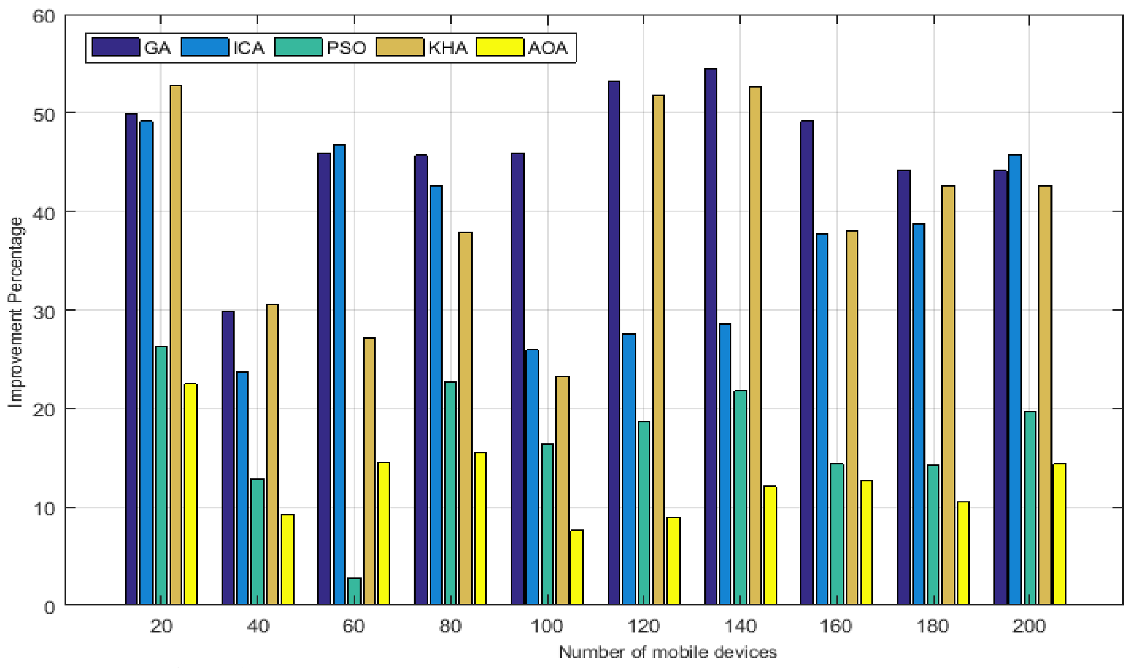


Figure 16. Comparison of BQAOA with other methods in terms of ESV rate improvement

8. Conclusion

The growing speed of diverse applications and the increasing demands for powerful computing platforms have been raised as a challenge. One of the reasons for the emergence of edge computing is a straightforward and quick access to powerful computing resources, decreased response time and enhanced user satisfaction. Due to limitations of computational resources on the edge and implementation of users' favorite computational application with a minimum energy consumption and quick processing time, the offloading decision is needed to increase the performance.

In this study, relying on MEC as an architectural concept of edge computing and offloading issue as an NP-complete problem, a new binary approach has been proposed based on AOA for offloading computational tasks to appropriate and efficient computational resources in terms of energy and response time. Two different scenarios have been used to evaluate the proposed offloaded scheme. The results showed that in the first scenario, BQAOA reduced the average energy consumption by about 3.94%, 6.58%, 15.64%, 20%, and 16.06% compared to the AOA, PSO, ICA, GA, and krill herd algorithms, respectively. The makespan time also decreased by about 4.6%, 5.7%, 12.8%, 16.52%, and 11.86% compared to the AOA, PSO, ICA, GA, and krill herd algorithms, respectively. Regarding the ESV metrics, BQAOA improved about 12.77% compared to the PSO algorithm, 30.45% compared to the ICA algorithm, 39.86% compared to the genetic algorithm, and 29.81% compared to the genetic with the krill herd algorithm, and 8.7% compared to the AOA algorithm. The results showed that, in the second scenario, BQAOA reduced the average energy consumption by about 3.28%, 5.28%, 18.64%, 22.7%, and 17.24% compared to the AOA, PSO, ICA and GA, and krill herd algorithm, respectively. The Makespan time also decreased by about 9.21%, 11.2%, 16.3%, 18.3%, and 19.7% compared to the AOA, PSO, ICA, GA, and krill herd algorithms, respectively. Regarding ESV metrics, BQAOA improved by about 17.17% compared to the PSO algorithm, 37.91% compared to the ICA algorithm, 45.05% compared to the genetic algorithm, and 40.12% compared to the genetic and krill herd algorithms, and 12.79% compared to the AOA algorithm.

In future studies, the researchers can introduce a multi-objective version of the BQAOA algorithm and consider different factors such as trust and security-related ones in solving the offloading problem.

References

[1] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 27, no. 2, pp. 1406-1427, 2019.

[2] O. H. Ahmed, J. Lu, A. M. Ahmed, A. M. Rahmani, M. Hosseinzadeh, and M. Masdari, "Scheduling of scientific workflows in multi-fog environments using Markov models and a hybrid Salp swarm algorithm," *IEEE Access,* vol. 8, pp. 189404-189422, 2020.

[3] M. Masdari and H. Khezri, "Efficient offloading schemes using Markovian models: a literature review," *Computing,* vol. 102, no. 7, 2020.

[4] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing,* vol. 23, no. 4, pp. 2399-2424, 2020.

[5] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal,* vol. 6, no. 3, pp. 5853-5863, 2019.

[6] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Joint computation offloading and scheduling optimization of IoT applications in fog networks," *IEEE Transactions on Network Science and Engineering,* vol. 7, no. 4, pp. 3266-3278, 2020.

[7] X. Deng, Z. Sun, D. Li, J. Luo, and S. Wan, "User-centric computation offloading for edge computing," *IEEE Internet of Things Journal,* 2021.

[8] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied computing and informatics,* vol. 14, no. 1, pp. 1-16, 2018.

[9] Z. Kuang, Z. Ma, Z. Li, and X. Deng, "Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing," *Journal of Systems Architecture,* vol. 118, p. 102167, 2021.

[10] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials,* vol. 19, no. 3, pp. 1628-1656, 2017.

[11] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*: Springer, 2018, pp. 103-130.

[12] S. Gharehpasha, M. Masdari, and A. Jafarian, "Virtual machine placement in cloud data centers using a hybrid multi-verse optimization algorithm," *Artificial Intelligence Review,* vol. 54, no. 3, pp. 2221-2257, 2021.

[13] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective," *Journal of Grid Computing,* vol. 18, no. 4, pp. 639-671, 2020.

[14] G. Peng, H. Wu, H. Wu, and K. Wolter, "Constrained Multi-objective Optimization for IoT-enabled Computation Offloading in Collaborative Edge and Cloud Computing," *IEEE Internet of Things Journal,* 2021.

[15] M. Huang, Q. Zhai, Y. Chen, S. Feng, and F. Shu, "Multi-Objective Whale Optimization Algorithm for Computation Offloading Optimization in Mobile Edge Computing," *Sensors,* vol. 21, no. 8, p. 2628, 2021.

[16] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*: Springer, 2019, pp. 43-55.

[17] D. Costa, "A tabu search algorithm for computing an operational timetable," *European Journal of Operational Research,* vol. 76, no. 1, pp. 98-110, 1994.

[18] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*: Springer, 1987, pp. 7-15.

[19] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research,* vol. 24, no. 11, pp. 1097-1100, 1997.

[20] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine,* vol. 1, no. 4, pp. 28-39, 2006.

[21] S. I. Gass and C. M. Harris, "Encyclopedia of operations research and management science," *Journal of the Operational Research Society,* vol. 48, no. 7, pp. 759-760, 1997.

[22] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Computer methods in applied mechanics and engineering,* vol. 376, p. 113609, 2021.

[23] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems,* vol. 12, no. 4, pp. 373-397, 2018.

[24] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications,* vol. 66, no. 4, pp. 1594-1608, 2017.

[25] X. Xu, D. Li, Z. Dai, S. Li, and X. Chen, "A heuristic offloading method for deep learning edge services in 5G networks," *IEEE Access,* vol. 7, pp. 67734-67744, 2019.

[26] R. Singh, S. Armour, A. Khan, M. Sooriyabandara, and G. Oikonomou, "Heuristic Approaches for Computational Offloading in Multi-Access Edge Computing Networks," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020: IEEE, pp. 1-7.

[27] I. B. Lahmar and K. Boukadi, "Resource allocation in fog computing: A systematic mapping study," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020: IEEE, pp. 86-93.

[28] Q.-V. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access,* vol. 6, pp. 75868-75885, 2018.

[29] C. Swain *et al.*, "Meto: Matching theory based efficient task offloading in iot-fog interconnection networks," *IEEE Internet of Things Journal,* 2020.

[30] L. N. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Applied Sciences,* vol. 10, no. 1, p. 203, 2020.

[31] N. I. M. Enzai and M. Tang, "A heuristic algorithm for multi-site computation offloading in mobile cloud computing," *Procedia Computer Science,* vol. 80, pp. 1232-1241, 2016.

[32] S. Guan, A. Boukerche, and A. Loureiro, "Novel Sustainable and Heterogeneous Offloading Management Techniques in Proactive Cloudlets," *IEEE Transactions on Sustainable Computing,* vol. 6, no. 2, pp. 334-346, 2020.

[33] E. V. D. Subramaniam and V. Krishnasamy, "Energy aware smartphone tasks offloading to the cloud using gray wolf optimization," *Journal of Ambient Intelligence and Humanized Computing,* pp. 1-9, 2020.

[34] M. Adhikari, S. N. Srirama, and T. Amgoth, "Application offloading strategy for hierarchical fog environment through swarm optimization," *IEEE Internet of Things Journal,* vol. 7, no. 5, pp. 4317-4328, 2019.

[35] V. Sundararaj, "Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm," *Wireless Personal Communications,* vol. 104, no. 1, pp. 173-197, 2019.

[36] P. COMPUTING, "An efficient job sharing strategy for prioritized tasks in mobile cloud computing environment using acs-js algorithm," *Journal of Theoretical and Applied Information Technology,* vol. 97, no. 4, 2019.

[37] T. Wang, X. Wei, C. Tang, and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints," *Peer-to-Peer Networking and Applications,* vol. 11, no. 4, pp. 793-807, 2018.

[38] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing,* 2019.

[39] Y. Zhang and J. Fu, "Energy-efficient computation offloading strategy with tasks scheduling in edge computing," *Wireless Networks,* vol. 27, no. 1, pp. 609-620, 2021.

[40] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *The Journal of Supercomputing,* vol. 71, no. 8, pp. 3009-3036, 2015.

[41] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 2, pp. 2092-2104, 2019.

[42] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proceedings of the ACM Symposium on Cloud Computing*, 2018, pp. 401-411.

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience,* vol. 41, no. 1, pp. 23-50, 2011.

[44] J. J. Moghaddam and A. Bagheri, "Suppressing vibration in a multilayers composite material plate using quantum-behaved particle swarm optimization and sliding mode control system," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering,* vol. 229, no. 11, pp. 2095-2105, 2015.

[45] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation,* vol. 6, no. 1, pp. 58-73, 2002.

[46] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys (CSUR),* vol. 52, no. 1, pp. 1-23, 2019.

[47] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in Fog computing using an improved NSGA-II," *Wireless Personal Communications,* vol. 102, no. 2, pp. 1369-1385, 2018.

[48] M. Masdari and M. Zangakani, "Green cloud computing using proactive virtual machine placement: challenges and issues," *Journal of Grid Computing,* pp. 1-33, 2019.

[49] M. Masdari, S. Gharehpasha, M. Ghobaei-Arani, and V. Ghasemi, "Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions," *Cluster Computing,* pp. 1-31, 2019.

[50] M. Masdari and M. Zangakani, "Efficient task and workflow scheduling in inter-cloud environments: challenges and opportunities," *The Journal of Supercomputing,* vol. 76, no. 1, pp. 499-535, 2020.

[51] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "Optimizing Computation Offloading in Satellite-UAV-Served 6G IoT: A Deep Learning Approach," *IEEE Network,* vol. 35, no. 4, pp. 102-108, 2021.

[52] L. N. Huynh, et al, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Applied Sciences* vol. 10.1, p. 203, 2020.