Offloading of mobile – cloud computing based on Lion Optimization Algorithm (LOA)

Original Scientific Paper

Ammar Mohammed Khudhaier*
Mustansiriya University, College of Science, Department of Computer Science
Baghdad, Iraq
promail2010@uomustansiriyah.edu.iq

Jamal Nasir Hasoon
Mustansiriya University, College of Science, Department of Computer Science
Baghdad, Iraq
jamal.hasoon@uomustansiriyah.edu.iq

*Corresponding author

Abstract *– Many problems and limitations appeared with the spread of mobile cloud computing (MCC) technology, due to the battery life of the mobile, limited resources, storage space and processors. In addition, the biggest challenge is to make decisions about executing tasks between local devices and cloud servers (edge) to save energy and reduce delay time by reducing the energy consumption (or cost) of the system concerned. In this paper, an optimization method based on the Lion Optimization Algorithm (**LOA**) is proposed for task- offloading of mobile cloud computing. Task- offloading is the transfer of tasks from mobile devices to servers that handle high-level computational operations, such as cloud servers or Mobile-Edge Computing (MEC) servers, to reduce task execution time and energy consumption. The proposed algorithm has a high performance in reducing the cost for offloading tasks. This method can be used to improve the performance of 5G, 6G and later mobile technologies to implement applications that require a large amount of computing, also, multi-objectives could be applied for more cost reduction. The results showed that the proposed algorithm **LOA** is better than other traditional algorithms such as the genetic algorithm in terms of calculating the costs to find the best solutions.*

Keywords*: task offloading, cloud servers, mobile-edge computing, mobile-cloud computing*

Received: April 26, 2025; Received in revised form: N/A; Accepted: July 14, 2025

1. INTRODUCTION

Due to rapid progress and development of communication and internet technology, and the widespread adoption of scalable, fault-tolerant, and highly available cloud computing technology led to the emergence of diverse applications used by many users across millions of edge devices which produces a big data [1], therefore request resources allocation and services in cloud computing with lack of coordination in resource utilization causes delays in internet services [2, 3]. Therefore, it is necessary to distribute cloud computing resources equally, ensuring high performance for internet services. This is achieved by load balancing on the all-cloud computing servers without burdening any server, improving resource utilization and reducing response time [4, 5]. Load balancing is a major challenge, requiring the use of unique hardware and software that consume high energy, leading to an increase in operating cost, as well as handling numerous applications that require rapid execution [6, 7].

The traditional methods used to improve performance do not provide the most appropriate solution to these devices and applications, therefore, over the past few years, researchers have developed many meta-heuristic algorithms to solve these problems [8], to reach algorithms with satisfactory performance, including algorithms inspired by nature [9]. An example on these algorithms, the algorithm inspired by improving the colony of ants searching for food [10] and the particle swarm optimization algorithm that simulates the social behavior of migratory birds searching for an unknown destination [11] and the bacterial food search algorithm that simulates the search for the best food for bacteria [12] and other algorithms. However, after analyzing these algorithms, one of the defects was revealed, which is in the rate of convergence of explorations or independence [5]. In 2012, Rajakumar and Ramakrishnan and Sankara Gomathi 2017 were able to mathematically design and formulate the Lion Optimization Algorithm (LOA) [13], which is a class of

powerful inferential models inspired by the solitary and cooperative behaviors of lions that differ from other algorithms, which has proven high performance in implementation [14]. The working principle of the lion optimization algorithm is based on defense and territory reserve with the aim of finding and replacing the worst solutions with the best ones, where the strongest pride lion shows its dominance compared to the territorial lion that will migrate or die [15]. The strongest pride lion represents the global minimum solution, and the territorial lion represents the local minimum solution. A group of lionesses hunts the prey by surrounding it from several axes and quickly attacks it through organized group hunting to succeed in hunting and continue life. Lions show other behaviors such as the method of hunting alone, placing territorial marks, and the difference between residents and nomad lions, as well as migration [9, 14].

2. RELATED WORK

Since the mobile cloud computing system has been used, many researchers have worked on finding optimal solutions to offload tasks to the cloud by reducing the energy consumed by the cloud infrastructure and maximizing resources by reducing the processing time of user tasks [16-18]. Accordingly, many nature-simulation algorithms have been proposed, including the proposed lion optimization algorithm, which is one of the Swarm Intelligence (SI) algorithms to solving optimization problems [19]. It has been employed in this paper for the process of tasks offloading [20], which have not been used for this purpose in the past. Some of the algorithms used for mobile cloud computing offloading are Chirag et.al. 2024 [21] Proposed an algorithm that combines elements of both the Chimpanzee and Whale Optimization Algorithms (CWOA). In the population initialization phase, to ensure that the population is evenly distributed over the solution space, a Sobol sequence is used, which increases the accuracy of the algorithm. The most important features of the chimpanzee-whale optimization algorithm are the neglect of false positives and the increase in computational speed. Several metrics are used to evaluate the efficiency of the algorithm, most notably task duration, delay, energy usage, and cost. Shuyue et.al. 2021 [22] Proposed the Particle Swarm Optimization (PSO) algorithm that is based on queueing. In the optimization process, Pareto optimality relationship defines the mobility probability to obtain the optimal solution. PSO has proven that the task offloading strategy balances energy consumption and reduces the server MEC delay as well as efficient resource allocation. Lina et.al. 2024 [23] Proposed the Ant Colony Optimization (ACO) algorithm that mimics the behavior of ants in searching for the shortest path to finding food and housing, by leaving pheromone trails that strengthen over time as they are used more. ACO reduces energy consumption and response time by searching for the most efficient paths. The experimental results of this algorithm demonstrated the methodology's effectiveness, significant improvements, and high-quality performance compared to traditional methods. Manal et.al. 2023 [24] Proposed the Binary Cuckoo Search (BCS) algorithm that mimics the behavior of cuckoos that replace weaker eggs with stronger ones by laying their eggs outside their nests, where the algorithm replaces less efficient solutions with better ones. Offloading decisions depend on several parameters, the most important of which are bandwidth and the number of tasks and mobile devices interacting with the edge server. The priority of each task is taken into account when making decisions. The binary cuckoo search algorithm is very efficient in reducing execution time when there are many mobile devices. Mohammad et.al. 2022 [25] Proposed Bacterial Foraging Optimization (BFO) algorithm, it is a nature-inspired algorithm which is based on multi-objective bacterial search optimization. The BFO algorithm is evaluated in comparison with the ant colony optimization (ACO), particle swarm (PSO) and RR algorithms, as it has proven superior to these algorithms in terms of communication cost, response time, and load management effectiveness.

3. LION OPTIMIZATION ALGORITHM (LOA)

Lions are a unique species of cat, characterized by their cooperative and competitive social relationships. There are two types of lions: resident lions, which live in groups called prides, and nomadic lions, which live separately and travel individually or in pairs. There are many behaviors that lions follow, the most important ones were chosen and represented mathematically to simulate nature [9, 26], which are as follows:

First behavior (hunting), Females hunt in three subgroups: two wings (left and right), and the central group. They move toward the prey, surrounding it from several directions. To simulate this behavior, the positions of both the prey and the hunters are updated. Initially the prey is dummy prey and is defined by Equation 1, and its position is updated by Equation 2. The positions of the hunters on the wings are updated by Equation 3, and those of the central hunters are updated by Equation 4.

$$PREY = \sum \frac{hunters(x1, x2, x3, \ldots, xn)}{the\ number\ of\ hunters} \quad (1)$$

Where $PREY$ is the position of dummy prey in the center of hunters, and hunters(x_1, x_2, …, x_n) is the summation position of hunters that will hunt.

$$PREY' = PREY + rand(0,1) \times PI \times (PREY - Hunter) \quad (2)$$

Where $PREY'$ is the new position of prey, $PREY$ is the current position of prey, $rand(0,1)$ is to add a random value between 0 and 1 to make the process of updating lion position more diverse and random, PI is the percentage of improvement in cost of hunter, and $Hunter$ is the new position of hunter who attack to prey.

$$Hunter' = \begin{cases} rand(Hunter, PREY), & Hunter < PREY \\ rand(PREY, Hunter), & Hunter > PREY \end{cases} \quad (3)$$

Where $Hunter'$ the new position of hunter who attacks prey, $Hunter$ is the new position of center hunter, and $PREY$ is the current position of prey.

$$Hunter' = \begin{cases} rand(2 \times PREY - Hunter, PREY), & (2 \times PREY - Hunter) < PREY \\ rand(PREY, 2 \times PREY - Hunter), & (2 \times PREY - Hunter) > PREY \end{cases} \quad (4)$$

Where $Hunter'$ is the new position of left or right hunter, and $Hunter$ is the current position of left or right hunter.

Second behavior (moving toward safe place), the remaining females move towards one of the territory's areas to search for the best one and record it as the best position visited. The females' positions are updating by equation (5).

$$Female\ Lion' = Female\ Lion + 2D \times rand(0,1)\{R1\} + U(-1,1) \times \tan(\theta) \times D \times \{R2\} \quad (5)$$

Where $Female\ Lion'$ is the new position of female lion, $Female\ Lion$ is the current position of female lion, D is the distance between the female lion's position and the selected point chosen by tournament selection among the pride's territory, U(-1, 1) is the uniformly distributed random number is used to prevent female lions from moving in a repetitive manner while moving to a safe place, $\{R1\}$ is a vector which its start point is the previous location of the female lion, and its direction is toward the selected position, $\{R2\}$ is the perpendicular to $\{R1\}$, and $tan(\theta)$ is the angle tangent is used to adjust the random distribution of female lions' movement while moving toward safe place.

Third behavior (Roaming) To search for the best position and record them as the best position visited, the resident males and the nomad males and females are roaming each according to their region. The positions of the resident males are updating by equation (6) and the nomad lions by equation (7).

$$x \sim U(0,2 \times d) \quad (6)$$

Where x is a random number with a uniform distribution that represents the amount of random movement of resident male lions while roaming, U(0,2×d) is a random number generated by uniform distribution that controls the amount of change as the lions' roam, and d is the distance between the male lion's position and the selected area of territory.

$$Lion_{ij}' = \begin{cases} Lion_{ij}, & if\ rand_j > pr_i \\ RAND_j, & otherwise \end{cases} \quad (7)$$

Where $Lion_{ij}'$ is the new position of ith nomad lion, j is dimension, $Lion_{ij}$ is the current position of ith nomad lion, j is dimension, $rand_j$ is a uniform random number within [0, 1], to check transition, $RAND_j$ is random generated vector in search space to generate new random location for lion, and pr_i is the probability of moving for each nomad lion independently.

Fourth behavior (Mating) Resident female mate with one or more resident males, while nomad female mate with only one nomad male. Each type of resident and nomad lions produces two cubs by equations (8) and (9).

$$\text{Offspring}_j 1 = \beta \times Female\ Lion_j + \sum \frac{(1-\beta)}{\sum_{i=1}^{NR} S_i} \times MaleLion_j^i \times S_i \quad (8)$$

$$\text{Offspring}_j 2 = (1-\beta) \times Female\ Lion_j + \sum \frac{\beta}{\sum_{i=1}^{NR} S_i} \times MaleLion_j^i \times S_i \quad (9)$$

Where $Offspring_j 1$ is the first cub, produced from mating parent female lion and male, $Offspring_j 2$ is the second cub, produced from mating parent female lion and male, β is a random number with a normal distribution with mean value 0.5 and standard deviation 0:1, to control the diversity of genetic traits, $Female\ Lion_j$ is the current position of female lion, $MaleLion_j^i$ is the current position of male lion, S_i is represent 1 for select male i for mating, and 0 for not select the male i, and NR is the number of resident males in a pride.

Fifth behavior (Defense) resident males defend the pride against new mature males and nomad males. If the resident lion is weaker than the new mature or nomad lion will either be they killed or migrating to become nomad.

Sixth behavior (Migration) to enhance population diversity and exchange of information between prides, resident females migrate from one pride to another or change their lifestyle to become nomad or vice versa.

4. PROPOSD METHOD

The proposed method relies on AI to solve the problem of offloading tasks from edge devices (mobile devices) to the cloud computing, this problem is NP-Hard problem [27, 28]. This method organizes the execution of operations consisting of multiple tasks by balancing these executions, some of which are on the edge devices and others on the cloud associated with the application executed in mobile applications that contain a cloud server. This is done by generating a set of random solutions called population and then selecting the best among them. The best solution, which represents the execution decision, depends on the cost resulting from a combination of energy consumption and latency. The lower the cost, the less energy in consumed to execute a task in the shortest time. This is achieved by applying the Lion Optimization Algorithm (LOA), which is a meta-heuristic algorithm [29], and which is represented by the diagram in (Fig. 1).

4.1. INITIAL POPULATION

As mentioned earlier in this paper, the Lion Optimization algorithm manages and organizes the offloading of tasks between the cloud and mobile devices. The algorithm first prepares the initial data for the tasks arriving from the cloud server. Let's assume we have a set of tasks to be executed, say ten, with each task divided among twelve workers. A decision is then made for ex-

ecuting tasks by each worker on the cloud (called the edge server) or on the mobile device (called the local device), based on the energy consumption of executing each task (cost). The proposed algorithm generates a set of random solutions, say 80 solutions. Each solution is a two-dimensional matrix consisting of 10 rows representing the tasks and 12 columns representing the workers who will execute the tasks. The values of these rows and columns are randomly generated zeros and ones. Ones represents the task will execute by the worker on the cloud, and zero on the local or mobile device. Table 1 is an example of one solution.

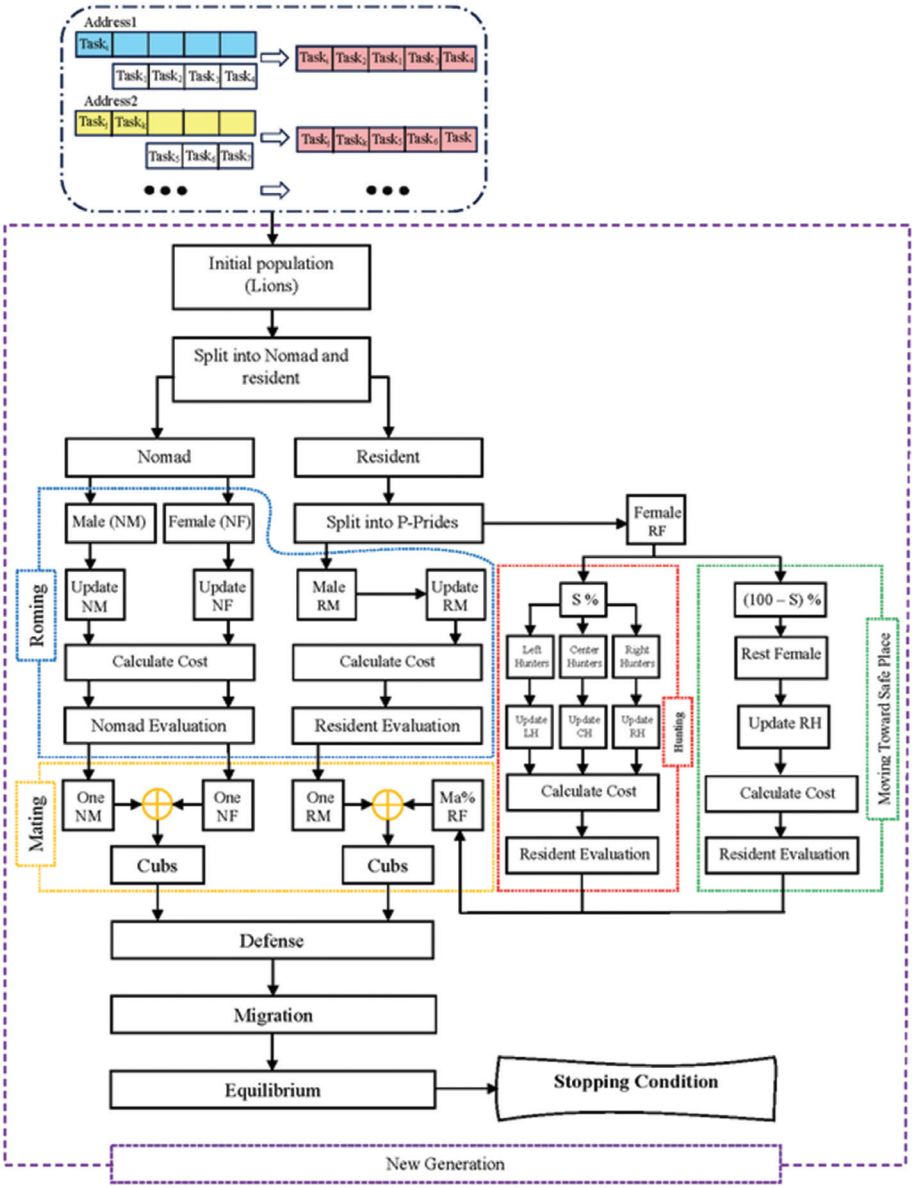

Fig. 1. Diagram of Lion Optimization Algorithm (**LOA**)

Table 1. Example of one solution for population

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	0	1	1	0	0	0	1	0	0	1	0
1	0	0	0	1	1	0	1	1	1	1	1	0
2	0	1	1	0	0	0	0	0	1	0	0	1
3	1	1	0	1	0	0	1	0	1	1	0	0
4	1	0	1	0	0	1	1	1	1	1	0	0
5	1	1	1	1	0	0	1	0	1	0	1	1
6	0	0	1	1	1	1	0	0	1	1	1	1
7	0	1	1	1	0	0	0	0	0	1	1	0
8	0	1	1	1	0	1	0	1	0	1	0	1
9	0	1	0	1	1	1	0	0	1	0	1	1

The set of solutions is called a population, and each solution is called a lion (male or female). Each lion (solution) is evaluated by calculating the cost of each solution to perform tasks based on initial values calculated mathematically.

For example, When calculating the cost of the solution in the Table 1, the result is (12923157959.379).

4.2. LIONS SPLITING

Lions are raandomly split into nomads and resident: nomads represent N% of the total population, and the rest are resident lions.

4.3. THE RESIDENT LIONS ARE SPLIT INTO P-PRIDES

Each pride is randomly split into females (lionesses) S% (between 75-90) of each pride, and the rest are males. The proportion of females in nomad lions is the inverse of the proportion in resident lions, i.e., 1-S, and males are the rest of nomad lions.

4.4. HUNTING

Hunters are females. Through the lion optimization algorithm, they are randomly divided into three groups: the center, which is the best solution, and the left and right wings. The algorithm selects one hunter at a time, and their positions are updated according to the group to which the hunter belongs. If the hunter belongs to the center group, their position is updated according to Equation (3), and Equation (4) if they are on either the left or right wings. Then, the new cost for each location is calculated and the costs are evaluated. If the cost of the new position is better (lower) than the cost of the previous position, the new cost is stored with its position in the list of best solutions, the prey'szposition is also updated according to Equation (2), whose position was previously calculated using Equation (1), but, if the previous position of hunter is better than the new position, the hunter is moved to another location.

Example 1: Hunting
Asumed the same solution example in the Table 1 is the current selected hunter with the following values:
Current hunter's position = 35 in the center group,
Current hunter's cost = **12923157959.379**
Current prey' position = 12
The **LOA** calculates the following **output** of new values:
New hunter's position = 17, New hunter's cost = **12552743255.394**
In this case hunter improved its own position and **LOA** will select the lowest cost for new position and update prey's position equal to 11.

4.5. MOVE TOWARD SAFE PLACE

After select female hunters for hunting, the rest females (100-S) from each pride move toward safe place. The LOA updating the position to get the new position by the equation (5), then calculate the cost of new position and evaluatting the cost of new position and previous position to save the best cost with it position in the best cost list.

Example 2: Move toward safe place
Asume the females number of pride = 9, The number of hunters = 7, then the number of rest female in the pride = 2 wich represent the females which are moving towared safe place.
After applying **LOA**, the solution at the current position (35), which represents (*Female Lion*) according to equation (5), its cost (**12923157959.379**) is computed using a fitness function.
After execution equation (5), a new solution (represent*Female Lion'* according to equation (5)) is created and its cost is computed.
In this example, the new cost is equal to (**12220873013.407**) which is less than the cost at the current position (35). Therefore, the LOA replaces the new solution with the currrent one at location 35.

4.6. ROAMING

During the roaming process of resident lions, the LOA randomly selects specific places within the pride's territory at a rate of R% based on the number of males. Each time, a male is selected from the resident lions to move from one place to another in search of the best solution. Using Equation (6), the lion's position is updated to calculate its cost and evaluate it with the cost of the previous position visited by each lion. As for the nomad lions, the lions are selected one by one, males and females, and their positions are updated according to Equation (7), then the costs are calculated and evaluated in comparison with the costs of the previous positions, then the best is chosen and stored in the list of the best solutions.

Example 3: lions roaming
Let's assume the following values:
$R = 90\%$
Number of males in the pride = 5
Locations to be selected from the pride's territory = $5 * R = 4$
The lion selected for roaming = 51
The cost of lion selected = **145776.230**
The **LOA** updating these values to be the following output results:
The cost of new position = **221497.173**
The **LOA** don't store the cost of new position, due to its lowest than cost of previous position.

4.7. Mating

To simulate the mating process among resident lions, the **LOA** selects a percentage (Ma%) of the total females in each pride to mate with one or more males, randomly selected by the **LOA**. In nomad lions, mating occurs between one female and one male. Based on Equations (8) and (9), two cubs are produced in both resident and nomad lions, whose sexes are randomly determined later. Both sexes are also genetically mutated at a rate of Mu%. The new cubs are then added to the population.

Example 4: lions mating
Population size = 100, number of prides = 4 for resident lions, the number of females to mate = 3
The output results:
• Each female mates with two males, producing two

cubs of lions. This means that there are 7 matings and two cubs per mating, producing 14 cubs for each pride of resident lions, as follows:
3 matings * 2 cubs * 4 prides = 24 cubs. In this example, this is for resident lions only.
However, in nomad lions, only two cubs are produced, which are added to the total number of cubs.
Total number of cubs = 26.
The final population size = 126 lions, or solutions.

4.8. DEFENSE

In the **first** instance of defense against new mature resident males, as mentioned above, the **LOA** first merges the positions of old males with new mature males, then sorts them in descending order of cost per position. Finally, the weakest male from the pride, the one with the highest cost, is removed from the list and becomes a nomad male. The **second** defense is against nomad male lions. **LOA** works by randomly selecting the prides to be attacked by nomad male (nomad male are selected one by one). The prides to be attacked are represented by a randomly selected list of zeros and ones. One means the pride will be attacked, and zero means it will not attack. If the nomad male is less cost than the resident male, the resident male is drive out from the pride, becoming a nomad lion, and the nomad male becomes a resident lion.

Example 5: First instance of defense
old resident males of the first pride from four prides = [92, 43, 70, 84, 90]
Cubs males of the first pride = [100, 102, 104, 106, 108, 110, 112]
Nomad male lions = [11, 29, 69, 97, 35, 28, 36, 34, 51, 30, 14, 24, 89, 21, 13, 156]
Number = 16
Output
List of resident males after merging = [92, 43, 70, 84, 90, 100, 102, 104, 106, 108, 110, 112]
The same list arranged in ascending order according to costs =
[43, 100, 102, 104, 106, 84, 90, 92, 112, 70, 108, **110**]
After the drive out of the weakest lion, which is position **110**, the new list of resident lions becomes =
[43, 100, 102, 104, 106, 84, 90, 92, 112, 70, 108]
List of nomad lions after the weakest lion moves to it = [11, 29, 69, 97, 35, 28, 36, 34, 51, 30, 14, 24, 89, 21, 13, 156, **110**]
Final list of nomad lions after the four weakest resident males move = [11, 29, 69, 97, 35, 28, 36, 34, 51, 30, 14, 24, 89, 21, 13, 156, 110, 126, 138, 45]
Nomad males number = 20 lions or solutions

4.9. MIGRATION

To simulate the migration behavior of lions, the algorithm configures the number of females to migrate (*Number of migratory females* = [*Number of surplus females per pride + Number of resident females allowed per pride * I%*] * *Number of prides*). The surplus females represent the females produced during the mating among resident lions. After selecting the females designated for migration, the algorithm moves them to nomad females by removing them from the resident females list and adding them to the nomad females list. The nomad females list is sorted in ascending order according to the best female, i.e., from the lowest cost to the higher one. The number of females removed from the resident females list is replaced by the lowedt cost females from nomad females list as (*Number of females required to replace the resident females = Number of resident females allowed per pride – Number of migratory females per pride*).

4.10. EQUILIBRIUM

To equilibrium the number of lions in the population, the algorithm works by returning it to the allowed original population size (for example, 100 solutions or lions), by eliminating the solutions with the highest costs. If the population size reaches, for example, 120 solutions or lions, the 20 solutions with the highest costs are eliminated, meaning that 100 solutions with costs lower than the original 20 solutions are retained.

4.11. CONVERGENCE

In this paper, the search for the best solutions stops after 200 iterations are completed. The best solution is selected from among 100 solutions (100 solutions = population size) from each iteration resulting 200 best solutions, which are compared with other 200 best solutions generated by the Genetic Algorithm **GA** [30-32].

5. EXPERIMENTAL RESULT

For the purpose of evaluating **LOA** by comparing it with **GA** the same initial data is used, the result of costs in all tables (3, 4, 5) and all diagrams in the figures (7-10) are generated by the implementing each of the **LOA** based on single objective and Implementing **GA**. First, the data is described which is used for this purpose in section 5.1, then comparing the results for evaluating **LOA** through the cost values of tables in section 5.2 and diagrams in section 5.3 with fixed some values and changing others.

Example 7: one of the 12 cases.
[array([[0.000749, 0.00106, … , 0.000676, 0.000996],
[0.000551, 0.00049, … , 0.000807, 0.000801],
. .
. .
[0.000393, 0.000816, … , 0.000493, 0.00796],
[0.000513, 0.000729, … , 0.000633, 0.000353]]),
array([……..……..…]),
array([……..…..……]),
array([……..…..……])]

5.1. DATA DESCRIPTION

To evaluate **LOA** by comparing it with **GA**, randomly generated data was used which matching the real dataset that is currently unavailable. Twelve cases from this random data were recorded and saved in a PKL file in the form of a three-dimensional matrix. Each case contains four arrays, each one includes four variables (T_{local}, E_{local}, T_{edge}, *and* E_{edge}).

Where T_{local} is the local time consumption, E_{local} is the local energy consumption, T_{edge} is the marginal time consumption, and E_{edge} is the marginal energy consumption, all for the worker w_j task c_i, ($i \in 1, 2, 3, \ldots, n$) and ($j \in 1, 2, 3, \ldots, m$).. Each variable consists of a matrix of 10 rows and 12 columns, as in Table 2. The data of the four variables are created based on the r_{local} = 656725 (Local data processing rate), q_{local} = 356713 (Consumption per bit of data processed locally), r_{edge} = 127571 (The rate at which edge server data is processed), q_{edge} = 852433 (Transmission energy consumption per bit data of edge server), x_{edge} = 469182 (Edge server data transfer rate), and yedge = 114295 (Energy consumption per bit of data processed by edge servers).

5.2. EVALUATING LOA BY COMPARING IT WITH GA IN DEFERENT CASES

All cases for **LOA** have constant values which are the roaming (R) = 30%, resident females (S) = 75%, prides number (P) = 4, mating (Ma) = 20%, resident females' surplus (I) = 10%, and nomad (N) = 10%.

a. Mutation (Mu)= 3% and iteration= 100, Table 3.

b. Population size=200 and iteration= 100, Table 4.

c. Population size=200 and mutation (Mu)=3%, Table 5.

Table 2. Complete values of array for the variable T_{local}

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.00074	0.00106	0.00046	0.00096	0.00096	0.00062	0.00034	0.00076	0.00037	0.00106	0.00067	0.00099
1	0.00055	0.00049	0.00071	0.00064	0.00060	0.00069	0.00094	0.00051	0.00053	0.00091	0.00080	0.00080
2	0.00102	0.00044	0.00070	0.00040	0.00051	0.00106	0.00101	0.00084	0.00035	0.00103	0.00102	0.00080
3	0.00077	0.00047	0.00087	0.00105	0.00045	0.00081	0.00076	0.00088	0.00095	0.00096	0.00103	0.00046
4	0.00055	0.00060	0.00062	0.00046	0.00098	0.00069	0.00091	0.00084	0.00037	0.00036	0.00098	0.00083
5	0.00064	0.00087	0.00077	0.00059	0.00098	0.00035	0.00101	0.00088	0.00096	0.00043	0.00072	0.00046
6	0.00076	0.00105	0.00086	0.00070	0.00106	0.00080	0.00092	0.00073	0.00039	0.00099	0.00053	0.00071
7	0.00051	0.00058	0.00064	0.00103	0.00085	0.00080	0.00098	0.00104	0.00096	0.00106	0.00090	0.00041
8	0.00039	0.00081	0.00037	0.00047	0.00042	0.00092	0.00091	0.00033	0.00056	0.00077	0.00049	0.00079
9	0.00051	0.00072	0.00062	0.00065	0.00094	0.00066	0.00090	0.00058	0.00055	0.00103	0.00063	0.00035

Table 3. Three different Population sizes

	Population size, Mu = 3%, iteration = 100					
Data Case	**100**		**150**		**200**	
	LOA-cost	GA-cost	LOA-cost	GA-cost	LOA-cost	GA-cost
0	8723034721.51	9720584791.47	8578311175.52	10019971021.46	8372255875.52	10468444321.44
1	8132422510.49	9954436198.41	7793764564.50	9715896886.42	7522013986.51	9349845706.44
2	8985631163.52	10328626883.47	8794848197.53	10486441001.46	8245286591.55	10666072739.45
3	8505810941.52	10517637923.44	8387753375.52	10036923029.45	8026308137.53	9242034407.49

Table 4. Three different Mutation ratios

	Mutation, Population size= 200, iteration = 100					
Data Case	**3%**		**5%**		**7%**	
	LOA-cost	GA-cost	LOA-cost	GA-cost	LOA-cost	GA-cost
0	8372255875.52	10468444321.44	8434072465.52	10040818969.46	8657339443.51	9940942753.46
1	7522013986.51	9349845706.44	7886610658.50	9440267620.44	8218965736.48	9967284352.41
2	8245286591.55	10666072739.45	8459584103.54	10928369015.44	8774242667.53	10124510927.48
3	8026308137.53	9242034407.49	8662655387.51	9487361423.48	8826045119.50	10269644309.45

Table 5. Three different Iterations

	Iteration, Population size= 200, Mu = 3%,					
Data Case	**100**		**200**		**300**	
	LOA-cost	GA-cost	LOA-cost	GA-cost	LOA-cost	GA-cost
0	8372255875.52	10468444321.44	7698576253.55	9501923755.48	8044264321.54	10040576551.46
1	7522013986.51	9349845706.44	7922246104.50	9206576668.44	7944548560.49	9730441966.42
2	8245286591.55	10666072739.45	8475098855.54	9900516695.48	8398009931.54	10492259033.46
3	8026308137.53	9242034407.49	8371511369.52	9481785809.48	7868326601.57	9670387013.47

5.3. EVALUATING LOA BY DIFFERENT CURVES OF LOA WITH CHANGING SOME VALUES

The constant values are population size = 150, Iteration= 100, prides number (P) = 4, resident females (S) = 75%, mating (Ma) = 20%, mutation (Mu) = 3%, and resident females' surplus (I) = 10%.

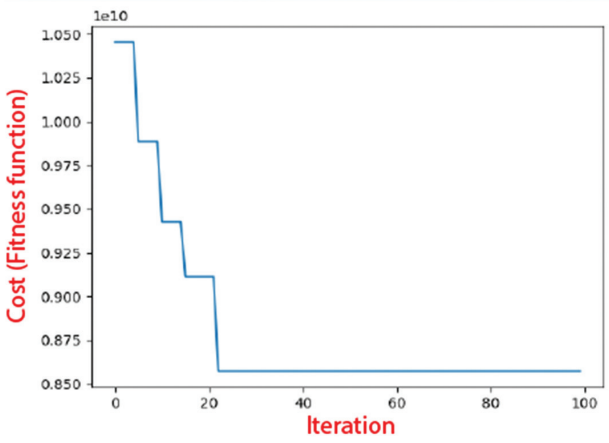

Fig. 2. $N = 10\%$, $R = 30\%$

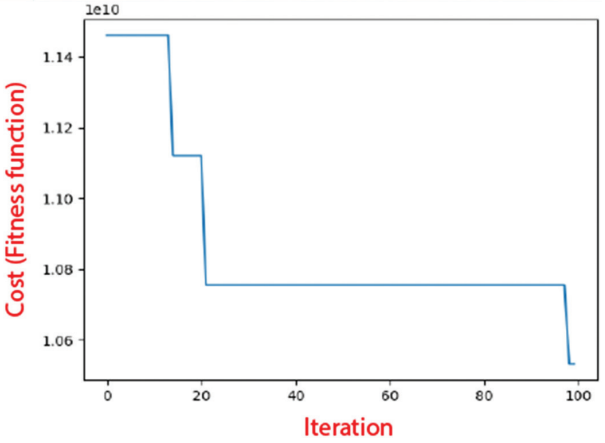

Fig. 3. $N = 10\%$, $R = 5\%$

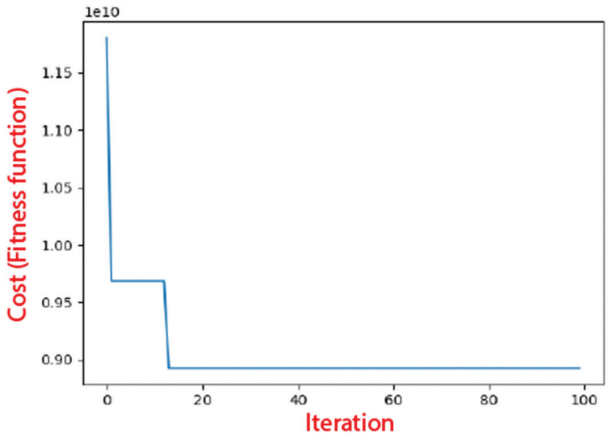

Fig. 4. $N = 40\%$, $R = 30\%$

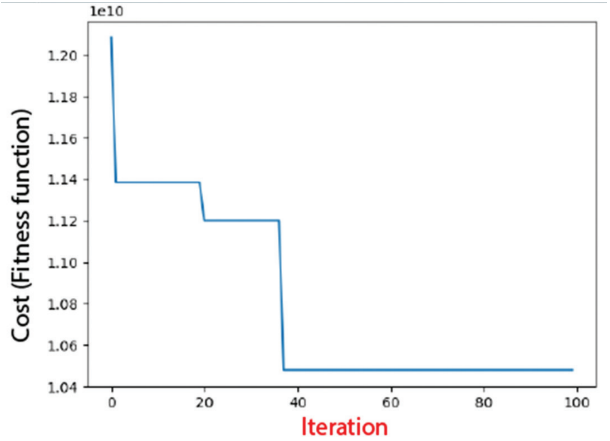

Fig. 5. $N = 40\%$, $R = 5\%$

6. CONCLOSION

In mobile cloud computing task-offloading, an optimization technique based on the **LOA** is suggested. In order to decrease task execution time and energy consumption, tasks are transferred from mobile devices to servers that manage complex computational operations, such as cloud servers or Mobile-Edge Computing servers. The suggested approach performs well in lowering offloading work costs. Multi-objectives could be utilized for further cost reduction. **LOA** has several groups, each of them attracted to produced new solutions that enhances the algorithm's ability to adapt in different environments. the experimental tests when increasing the mutation rates. The performance of **LOA** in offloading is butter than **GA** when applied in same sample data. The evaluation process shows through comparing the **LOA** with **GA** in the all tables that the results of the costs by **LOA** are always less than **GA**. For example, in the record (0) of Table 3, the costs of two algorithms are as follows: **LOA** cost = 8723034721.51, **GA** cost = 9720584791.47. The unit of cost is a combination of joules and second.

7. ACKNOWLEDGEMENT

The authors would like to thank Mustansiriyah University (www.uomustansiriyah.edu.iq Baghdad Iraq) for its support in the present work.

8. REFERENCES

[1] J. Hasoon, R. Hassan, "Big Data Techniques: A Survey", Iraqi Journal of Information Technology, Vol. 9, No. 4, 2019.

[2] M. Pai, S. Rajarajeswari, D. P. Akarsha, S. D. Ashwini, "Analytical Study on Load Balancing Algorithms in Cloud Computing", Expert Clouds and Applications, Vol. 209, 2022, pp. 631-646.

[3] S. Dong, J. Tang, K. Abbas, R. Hou, J. Kamruzzaman, L. Rutkowski, R. Buyya, "Task offloading strategies for mobile edge computing: A survey", Computer Networks, Vol. 254, 2024, p. 110791.

[4] M. Hashemi, A. Masoud, "Load Balancing Algorithms in Cloud Computing Analysis and Performance Evaluation", 2020, https://philarchive.org/archive/HASLBA (accessed: 2025)

[5] R. Kaviarasan, G. Balamurugan, R. Kalaiyarasan, Y. Reddy, "Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm", e-Prime - Advances in Electrical Engineering, Electronics and Energy, Vol. 6, 2023, p. 100326.

[6] J. Adaikalaraj, C. Chandrasekar, "To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm", Measurement: Sensors, Vol. 27, 2023, p. 100834.

[7] A. Gaikwad, K. Singh, S. Kamble, M. Raghuwanshi, "A comparative study of energy and task efficient load balancing algorithms in cloud computing", Physics: Conference Series, Vol. 1913, 2021.

[8] B. Gerald, D. R. P. Geetha, "Metaheuristic algorithm-based load balancing in cloud computing", Theoretical and Applied Information Technology, Vol.102, No. 5, 2024, pp. 2099-2115.

[9] M. Yazdani, F. Jolai, "Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm", Computational Design and Engineering, Vol. 3, No. 1, 2016, pp. 24-36.

[10] A. Alaidi, C. Soong Der, Y. W. Leong, "Systematic Review of Enhancement of Artificial Bee Colony Algorithm Using Ant Colony Pheromone", International Journal of Interactive Mobile Technologies, Vol. 15, No. 16, 2021, pp. 172-180.

[11] Q. You, B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things", Cloud Computing, Vol. 10, No. 41, 2021.

[12] C. Guo, H. Tang, B. Niu, C. P. Lee, "A survey of bacterial foraging optimization", Neurocomputing. Vol. 452, 2021, pp. 728-746.

[13] B. Rajakumar, "Lion Algorithm and Its Applications", Frontier Applications of Nature Inspired Computation, 2020, pp. 100-118.

[14] M. Mansor, M. Kasihmuddin, S. Sathasivam, "Modified Lion Optimization Algorithm with Discrete Hopfield Neural Network for Higher Order Boolean Satisfiability Programming", Malaysian Journal of Mathematical Sciences, Vol. 14, 2020, pp. 47-61.

[15] S. Almufti, "Lion algorithm: Overview, modifications and applications", international research journal of science, technology, education and management, Vol. 2, No. 2, 2022, pp. 176-186.

[16] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, S. Papavassiliou, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions", Computer Networks, Vol. 195, 2021, p. 108177.

[17] X. Chen, G. Liu, "Federated Deep Reinforcement Learning-Based Task Offloading and Resource Allocation for Smart Cities in a Mobile Edge Network", Sensors, Vol. 22, 2022, p. 4738.

[18] X. Shichao, Y. Zhixiu, W. Guangfu, L. Yun, "Distributed Offloading for Cooperative Intelligent Transportation Under Heterogeneous Networks", IEEE Transactions on Intelligent Transportation Systems, Vol. 23, No. 9, 2022, pp. 16701-16714.

[19] A. Al-Obaidi, H. Abdullah, Z. Ahmed, "Camel Herds Algorithm: a New Swarm Intelligent Algorithm to Solve Optimization Problems", International Journal on Perceptive and Cognitive Computing, Vol. 3, No. 1, 2017.

[20] L. Meng, Y. Wang, H. Wang, X. Tong, Z. Sun, Z. Cai, "Task offloading optimization mechanism based on deep neural network in edge-cloud environment", Cloud Computing, Vol. 12, No. 76, 2023.

[21] C. Chandrashekar, P. Krishnadoss, V. Kedalu, B. Anantharakrishnan, "MCWOA Scheduler: Modified Chimp-Whale Optimization Algorithm for Task Scheduling in Cloud Computing", Computers, Materials and Continua, Vol. 78, No. 2, 2024, pp. 2593-2616.

[22] M. Shuyue, S. Song, L. Yang, J. Zhao, F. Yang, L. Zhai, "Dependent tasks offloading based on particle swarm optimization algorithm in multi-access edge computing", Applied Soft Computing, Vol. 112, 2021, p. 107790.

[23] L. Ibrahim, O. Fadare, F. Al-Turjman, A. ALwhelat, "Ant Colony Optimization with Lagrangian Relaxation for Cloud Computing Offloading Optimiza-

tion", Dijlah Journal of Engineering Science, Vol. 1, No. 1, 2024, pp. 49-56.

[24] M. Alqarni, A. Cherif, E. Alkayyal, "ODM-BCSA: An Offloading Decision-Making Framework based on Binary Cuckoo Search Algorithm for Mobile Edge Computing", Computer Networks, Vol. 226, 2023, p. 109647.

[25] M. Babar, A. Din, O. Alzamzami, H. Karamti, A. Khan, M. Nawaz, "A Bacterial Foraging Based Smart Offloading for IoT Sensors in Edge Computing", Computers and Electrical Engineering, Vol. 102, 2022, p. 108123.

[26] S. Sridhara, A. Chakravarthy, U. Nandini, "Asiatic Lions: Behaviour in the Wild and Captivity", Animal Behavior in the Tropics, Springer, 2025, pp. 491-502.

[27] X. Tan, M. Emu, S. Choudhury, "Task Offloading in Mobile Edge Computing: Intractability and Proposed Approaches", Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Niagara Falls, ON, Canada, 17-20 November 2022, pp. 775-778.

[28] A. ALRIDHA, A. Salman, A. Al-Jilawi, "The Applications of NP-hardness optimizations problem", Journal of Physics: Conference Series, Vol. 1818, 2021, p. 012179.

[29] B. Vijayaram, V. Vasudevan, "Wireless edge device intelligent task offloading in mobile edge computing using hyper-heuristics", EURASIP Journal on Advances in Signal Processing, Vol. 2022, 2022, p. 126.

[30] Z. Li, Q. Zhu, "Genetic Algorithm-Based Optimization of Offloading and Resource Allocation in Mobile-Edge Computing", Information. Vol. 11, No. 83, 2020.

[31] H. Yektamoghadam, R. Haghighi, M. Dehghani, A. Nikoofard, "Genetic Algorithm and Its Applications in Power Systems", Frontiers in Genetics Algorithm Theory and Applications, Tracts in Nature-Inspired Computing, Springer, 2024, pp. 83-97.

[32] M. Younis, S. Yang, "A Genetic Algorithm for Independent Job Scheduling in Grid Computing", MENDEL Soft Computing, Vol. 23, No. 1, 2017.