# Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization

Mohamed H. Mousa[1,2,*] and Mohamed K. Hussein[2,*]

[1] Department of Information Technology, College of Computer Science at AlKamil, University of Jeddah, Jeddah, Saudi Arabia
[2] Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt
* These authors contributed equally to this work.

Corresponding author
Mohamed K. Hussein,
m_khamiss@ci.suez.edu.eg

## ABSTRACT

Internet of Things (IoT) tasks are offloaded to servers located at the edge network for improving the power consumption of IoT devices and the execution times of tasks. However, deploying edge servers could be difficult or even impossible in hostile terrain or emergency areas where the network is down. Therefore, edge servers are mounted on unmanned aerial vehicles (UAVs) to support task offloading in such scenarios. However, the challenge is that the UAV has limited energy, and IoT tasks are delay-sensitive. In this paper, a UAV-based offloading strategy is proposed where first, the IoT devices are dynamically clustered considering the limited energy of UAVs, and task delays, and second, the UAV hovers over each cluster head to process the offloaded tasks. The optimization problem of dynamically determining the optimal number of clusters, specifying the member tasks of each cluster, is modeled as a mixed-integer, nonlinear constraint optimization. A discrete differential evolution (DDE) algorithm with new mutation and crossover operators is proposed for the formulated optimization problem, and compared with the particle swarm optimization (PSO) and genetic algorithm (GA) meta-heuristics. Further, the ant colony optimization (ACO) algorithm is employed to identify the shortest path over the cluster heads for the UAV to traverse. The simulation results validate the effectiveness of the proposed offloading strategy in terms of tasks delays and UAV energy consumption.

## INTRODUCTION

Mobile edge computing (MEC) has emerged as a promising foundation for providing quality of service (QoS) requirements for the Internet of Things (IoT) and mobile devices to overcome the limited resource capabilities of these devices in terms of processing cycles and energy (*Hussein & Mousa, 2020*). MEC is an edge network of servers supported by a backend layer of cloud computing located near the IoT and mobile devices network. Offloading computational tasks of the IoT and mobile devices to the edge servers improves execution delay and power consumption of devices by taking advantage of low bandwidth

and high latency (*Wang et al., 2017*). However, there are certain locations where deploying edge servers could be difficult or even impossible, such as hostile terrain, deserts, mountains, underwater, wilderness areas, and disaster areas where the network is down (*Mohamed et al., 2017*; *Cheng et al., 2018*).

Unmanned aerial vehicles (UAVs) have significantly advanced in both technological aspects and cost aspects and have shown prominent success in distinct applications, including military, traffic control, farming and wilderness monitoring applications (*Bejaoui, Park & Alouini, 2020*). This outstanding success is motivated by the agility, mobility, and cost-effective deployment of UAVs (*Zhang et al., 2019*). UAVs can be employed in MEC architecture in two distinct ways: (1) UAV-assisted communication MEC architectures, and (2) UAV-based computation offloading MEC architecture. In the former approach, UAVs serve as relays for distant ground base stations, allowing fast, flexible, and cost-effective network coverage for IoT devices (*Wang et al., 2019*; *Fu et al., 2020*). In the latter approach, an edge server is mounted on a UAV for processing offloaded computation tasks of ground mobile devices. This integration of a UAV with an MEC network and the short distance line-of-sight (LoS) wireless communication between the UAV and wireless devices improves the QoS requirements of mobile applications in terms of delay sensitivity as well as the energy consumption of the wireless devices (*Mao et al., 2017*; *Zhou et al., 2020*). However, UAVs suffer from limited energy and computation capacity constraints, which affect the delay of offloaded tasks. These constraints represent a major challenge that needs to be addressed in UAV-based computation offloading MEC architectures.

In this paper, an intelligent, UAV-based, computation offloading strategy is investigated using evolutionary metaheuristics, where a single UAV-based edge server is utilized to provide computation offloading service to ground IoT and mobile devices. The proposed intelligent architecture aims to minimize the optimization objective of UAV energy and task delays. Two key challenges need to be addressed: (1) the deployment of the UAV, and (2) the shortest trajectory path for the UAV over the set of deployments coordinates. The IoT devices are partitioned into clusters, and the UAV hovers over each cluster head to process the offloaded tasks of the cluster members. A UAV shortest trajectory over the cluster heads is optimized.

UAV energy, deployment, UAV trajectory, and task delays are key challenges in several UAV-based, MEC architecture (*Wu et al., 2019*; *Li et al., 2020a*; *Wang et al., 2021*; *Fu et al., 2021*). However, approaches such as greedy search and exhaustive search cannot be applied to such optimization problems because there are many discrete and continuous decision parameters, including number, location of deployment, the offloading decision in each deployment, and the shortest trajectory path. Consequently, the complexity of these approaches is extensive and the execution time is extremely high. Further, deep learning approaches require training data, which may not be feasible in a mobile and stochastic environment. Therefore, the optimization problem is divided into two subproblem in order to find a near optimal solution in real-time for such nondeterministic polynomial-time (NP)-hard problems in an acceptable complexity and time. The first subproblem is clustering the IoT devices into groups where the UAV hovers over each

group for processing the offloaded tasks of each group members. This problem is an NP-hard complex optimization problem, and evolutionary algorithms can be applied to find a near optimal solution in an acceptable computation time (*Hu et al., 2019*, *2020*; *Wu et al., 2020*). Therefore, we investigate the use of evolutionary meta-heuristics, including differential evolution (DE), particle swarm optimization (PSO), and genetic algorithms (GAs), to obtain a near-optimal solution for the clustering problem considering energy and time. The second subproblem is the determination of the shortest UAV trajectory over the clusters head considering the time and energy of the UAV flying. The contributions of this paper are presented as follows:

- The distribution of mobile devices in a specific area is non-uniform. Therefore, partitioning and load balancing offloaded computing tasks into a set of regions severely impacts the performance of the offloading system. Performance degradation may occur, as some regions may become heavily loaded with requests while other regions are lightly loaded with requests. As a result, the waiting time increases in the heavily loaded regions, and some offloading requests could fail. In addition, the UAV energy may be lost if a large number of clusters is set. Therefore, the UAV deployment optimization problem is formulated as mixed-integer, nonlinear constraint optimization that is aimed at minimizing the number of clusters while considering the energy consumption of the UAV and the delays of the offloaded tasks.

- Two discrete differential meta-heuristics are proposed with mutation and crossover operators, namely, discrete DE (DDE) and discrete PSO (DPSO), for the formulated optimization problem. The proposed DDE algorithm is proposed to obtain a near-optimal solution for the number of clusters, members of each cluster and load balancing tasks in the determined clusters.

- The energy of the UAV is limited, and the UAV follows a determined trajectory over different regions to process the offloaded computational tasks and to return the results. Therefore, the UAV trajectory should be optimized to address the energy limitations considering the communication, computation, and mechanical operations of flying and hovering. The partitioned regions are employed as an initial population for an ant colony optimization (ACO) algorithm to obtain a near-optimal solution for minimizing the UAV trajectory.

- Several extensive experiments are performed with the proposed strategy. The evaluations show that the proposed offloading system is effectively capable of significantly improving the task delay and energy consumption compared with a discrete particle swarm optimization (DPSO) and the genetic algorithm (GA).

The remainder of the paper is organized as follows: "Related Work" introduces the related work of UAV-assisted, mobile edge environments. "UAV-Based Offloading System Model and Problem Formulation" presents a system model and a formulation of the optimization problem, while "Proposed Meta-Heuristics" outlines the proposed optimization using DDE, DPSO, and ACO. In "Experimental Results", the experimental results of the proposed strategy are presented and evaluated, followed by a summary and

conclusion of the paper with a discussion of future research directions in "Conclusion and Future Work".

## RELATED WORK

The MEC architecture is a successful solution for the limited computational resources in terms of computation and energy, and thus, the performance of the offloaded tasks are improved by taking advantage of the low latency and high bandwidth of the edge servers (*Yu, 2016*; *Zakaryia, Ahmed & Hussein, 2020*). UAVs have the advantages of agility, mobility and fast deployment and can provide on-demand communication and computation services when mounted with communication equipment and MEC servers with a LoS advantage, which will produce better transmission rates with reduced energy consumption of the mobile devices. However, the limitation of the energy of the UAV affects the communication and computational capacity, as well as the service time offered to the ground mobile equipment (*Nguyen et al., 2020*). Therefore, different research efforts are conducted for the key challenges based on the different configurations, optimization objectives, and underlying constrains.

In *Hu et al. (2019a)*, the UAV trajectory and ratio of offloading tasks are jointly optimized with the aim of minimizing the sum of the maximum delay of all users in different time slots, as statistically specified. In *Wang et al. (2020a)*, the task offloading decisions are reached based on the objective of minimizing offloading delay and energy consumption considering bandwidth, size of the data, and power consumption. The offloading decision for each mobile task includes offloading to the UAV or offloading to the ground MEC through the UAV. However, the study does not consider the optimization of UAV trajectories. In *Tang et al. (2020)*, a partial offloading UAV-based MEC system that is aimed at maximizing the number of offloading tasks is proposed. The offloading problem is formulated as a mixed-integer, nonlinear programming problem and is solved using block coordinate descent (BCD) and a convex optimization technique. However, the mobility of the UAV is not considered in the proposed model. In *Yang et al. (2020)*, DE with deep reinforcement learning is selected for the deployment of multi-UAVs with the aim of minimizing the offloaded task delays and load balancing UAVs loads. In *Li et al. (2020a)*, a successive complex optimization technique is utilized to jointly optimize the UAV trajectory and computational load allocation with the aim of minimizing UAV energy. However, the proposed system does not consider the distribution of the mobile devices on the ground area or the time requirements of the offloaded tasks. In *Wang et al. (2020b)*, a two-layer optimization method is proposed to jointly optimize multi-UAV deployment and mobile task allocation and to minimize UAV energy consumption. The upper layer of the optimization method uses a DE algorithm that minimizes the number of UAVs, and in the lower layer, a greedy algorithm is proposed for identifying an optimal solution whether tasks are offloaded or processed locally. In *Chen et al. (2020)*, an intelligent task offloading system is proposed. The proposed system intelligently perceives the network environment and makes offloading decisions using a Monte Carlo tree search. Furthermore, a deep neural network is applied to optimize the search based on the latency delay. However, the proposed offloading strategy requires

training data, a prediction model for the channel state, and time for the self-learning process. In *Hu et al. (2019b)*, a UAV is connected with a cellular base station as either a relay to the base station or as a computing server for the offloaded computation tasks of the mobile equipment. A greedy search based on nonconvex optimization is proposed with the aim of minimizing the weighted sum energy of the UAV and the mobile devices by jointly optimizing the computational resource scheduling, allocation of bandwidth resources, and trajectory of the UAV. In *Li et al. (2020b)*, an energy-efficient, UAV-based, offloading architecture that optimizes the bit allocations in different regions of user tasks and the trajectory of the UAV using successive approximation is proposed. However, the proposed scheme does not address how to partition the fixed clustered slots, and further, the time delay is not considered in the proposed scheme. In *Zhan et al. (2020)*, a joint optimization of computation offloading, resource allocation, and UAV trajectory is proposed with the aim of minimizing the energy consumption of the UAV. In *Guo & Liu (2020)*, the UAV energy is optimized considering the transmitted bits in both the uplink and downlink and the UAV trajectory using a greedy search based on the successive convex approximation. In *Yu et al. (2020)*, an alternative optimization algorithm based on successive convex approximation (SCA) is proposed to minimize the weighted sum of the service delay of the IoT tasks by jointly optimizing computing offloading, resource allocation and trajectory.

In summary, approaches such as greedy search and exhaustive search cannot be applied to such an optimization problem because there are many discrete and continuous decision parameters, including the number, location of deployment, offloading decision in each deployment, and shortest trajectory path. Therefore, most research uses heuristic approaches with fixed partitions; these approaches are not scalable because their complexity is extensive. Further, deep learning approaches, such as *Lan et al. (2019)*, *Wang et al. (2021)*, requires training data which may not feasible in the mobile and stochastic environment. Therefore, we opt to use evolutionary meta-heuristics, including DE, PSO, and GAs, to obtain a near-optimal solution for such NP-hard problems in an acceptable time. The proposed UAV-based offloading system determines the optimal number of clusters and specifies the member tasks of each cluster which are permitted to offload when the UAV hover the cluster head considering the UAV energy and task delay constraints.

## UAV-BASED OFFLOADING SYSTEM MODEL AND PROBLEM FORMULATION

A UAV-based offloading system consists of two layers. The first layer, the ground layer, is the IoT and mobile device layer, which has $N$ stationary wireless devices $S = \{s_1, s_2, \ldots, s_N\}$ on the ground. The wireless devices offload their computations to the edge layer to accelerate computations and to optimize energy consumption of the IoT devices. The second layer is the flying MEC layer, which consists of a single UAV that is supported with an edge server that offers computation offloading services for the ground layer with minimum latency. The UAV flies at a steady altitude $H > 0$. The UAV has the communication coverage radius $R_c$. The UAV flies according to a specified trajectory over a set of $K$ regions starting from a specific predetermined starting point. The UAV hovers

**Table 1  List of abbreviations.**

| Abbreviation | Definition |
|---|---|
| UAV | Unmanned arial vehicle |
| IoT | Internet of Things |
| QoS | Quality of service |
| MEC | Mobile edge computing |
| DE | Differential evolution |
| DDE | Discrete differential evolution |
| PSO | Particle swarm optimization |
| DPSO | Discrete particle swarm optimization |
| GA | Genetic algorithm |

over each region $k$ to process the offloaded tasks that are scheduled in this region, and the UAV returns to the starting point by the end of the trajectory cycle.

The coordinates of each $s_i$ device location in the ground layer are known in advance and are given by $s_i = \{x_i, y_i\}$. The coordinates of the UAV position in region $k$ are given by $u_k = \{x_k, y_k\}$, and the distance between the UAV and $s_i$ is calculated using the Euclidean distance:

$$dist_{ik} = [(x_k - x_i)^2 + (y_k - y_i)^2 + H^2]^{\frac{1}{2}} \tag{1}$$

where device $s_i$ should be in the cover radius $R_c$ of the UAV.

$$dist_{ik} < R_c \tag{2}$$

Device $s_i$ is requesting to offload task $t_i = \{t_i^c, t_i^\tau, t_i^{data}\}$, where $t_i^c$, $t_i^\tau$, and $t_i^{data}$ are the required computing cycles, task delay, and data size, respectively. In the proposed UAV-based offloading system, the set of ground devices $s_i \forall i \in N$ are partitioned into $K$ regions, where the UAV hovers over each region to process the offloaded tasks, and $\delta_{ik} = 1$ when mobile device $s_i$ offloads its task $t_i$ to be processed when the UAV hovers over region $k$. The following subsection presents the system formulation, followed by a formulation of the objective function. The key abbreviations used in the paper are listed in Table 1. The following subsections present the formulations for the computation, the communication, and the power consumption model for the optimization problem.

## Communication model

Assume that the UAV has a ground coverage range with a radius of $R_c$. Similar to *Bejaoui, Park & Alouini (2020)*, the time-varying channel gain between a ground IoT device $s_i$ within partition $k$ to the UAV *via* orthogonal frequency division multiplex access (OFDMA) is calculated using Eq. (3).

$$h_{ik} = \frac{\rho_0}{dist_{ik}^2} \tag{3}$$

where $\rho_0$ is the received power at a reference distance of 1 m.

The transmission rate between sensor $i$ and the UAV in region $k$ at position $u_k = \{x_k, y_k\}$ is calculated as (*He et al., 2018*) using Eq. (4).

$$r_{ik} = B\log_2\left(1 + \frac{P_i h_{ik}}{N_0}\right) \tag{4}$$

where $B$ is the bandwidth of the uplink channel, $P_i$ is the maximum transmit power of the wireless device $s_i$, and $N_0$ is the channel noise.

The transmission time $T_{ik}^{trans}$ for offloading a task from the $i^{th}$ IoT device to the UAV at region $k$ is calculated using Eq. (5).

$$T_{ik}^{trans} = \frac{t_i^{data}}{r_{ik}} \tag{5}$$

The total transmission time at region $k$ with position $\{x_k, y_k\}$ is calculated using Eq. (6).

$$T_k^{trans}(x_k, y_k, \delta_{ik_{i=1}}^N) = \sum_{i=1}^{N} \delta_{ik} T_{ik}^{trans} \tag{6}$$

where $\delta_{ik}$ represents the offloading decision for task $t_i$ in region $k$ with coordinates $\{x_k, y_k\}$, where $\delta_{ik} = 1$ means that the task is processed when the UAV hovers over region $k$.

## 3.2 Computation model

The computation time $T_{ik}^{comp}$ for processing task $t_i$ at the UAV is calculated using Eq. (7).

$$T_{ik}^{comp} = \frac{t_i^c}{f^u} \tag{7}$$

where $f^u$ is the processing capacity of the UAV.

Assume that the edge server executes the offloaded tasks in a queue ordered by first come first served. The overall computation time in region $k$ when the UAV hovers at position $\{x_k, y_k\}$ is calculated using the following equation:

$$T_k^{comp}(x_k, y_k, \delta_{ik_{i=1}}^N) = \sum_{i=1}^{N} \delta_{ik} T_{ik}^{comp} \tag{8}$$

The total time for the UAV in region $k$ is the total offloading delay, which is calculated as the sum of the transmission delay and computation delay using the following equation:

$$T_k(x_k, y_k, \delta_{ik_{i=1}}^N) = T_k^{trans}(x_k, y_k, \delta_{ik_{i=1}}^N) + T_k^{comp}(x_k, y_k, \delta_{ik_{i=1}}^N) \tag{9}$$

## Energy consumption model

The energy model which is used in this study follows the energy model described in *Bejaoui, Park & Alouini (2020)* for a fixed wing UAV. The UAV works under limited charged energy capacity, and the propulsion energy consumption is much higher than computation and communication energy (*Wang et al., 2020b*). The adopted model is simple where the energy is computed using a constant coefficient that depends on the

architecture. For example, the hovering energy is calculated by multiplying a hovering energy coefficient by the hovering time. The total UAV energy consumption is a result of several operations, including (1) wireless communication with IoT devices to receive offloaded tasks and to return results, (2) computation of the offloaded tasks, and (3) propulsion energy consumption of flying and hovering. The transmission energy consumption at region $k$ is calculated using Eq. (10).

$$E_k^{trans}(x_k, y_k, \delta_{ik_{i=1}}^N) = \kappa_1 \sum_{i=1}^{N} \delta_{ik} t_i^{data} \tag{10}$$

where $\kappa_1$ is an energy factor for wireless communication.

The computation energy consumption in region $k$ is calculated using Eq. (11).

$$E_k^{comp}(x_k, y_k, \delta_{ik_{i=1}}^N) = \kappa_2 \sum_{i=1}^{N} \delta_{ik} t_i^c f^{u2} \tag{11}$$

where $\kappa_2$ is an energy factor for computation processing.

Following the propulsion energy model of the UAV proposed in reference to the propulsion model proposed in *Wu et al. (2019)*, the propulsion energy consumption for hovering in region $k$ is calculated using Eq. (12).

$$E_k^h(x_k, y_k, \delta_{ik_{i=1}}^N) = \gamma T_k(x_k, y_k, \delta_{ik_{i=1}}^N) \tag{12}$$

where $\gamma$ is an energy factor for hovering and $T_k(x_k, y_k, \delta_{ik_{i=1}}^N)$ is the time in which the UAV hovers in region $k$ to process the offloaded tasks, calculated using Eq. (9).

The propulsion energy consumption for flying to region $k$ is calculated using Eq. (13).

$$E_k^{fly}(x_k, y_k) = \Gamma \frac{||u_k - u_{k-1}||^2}{\mu} \tag{13}$$

where $\Gamma$ is an energy factor for flying and $\mu$ is the UAV speed between the coordinates of regions $u_{k-1}$ and $u_k$.

The total energy consumption of the UAV is the sum of the resulting transmission energy consumption, resulting computing energy, and propulsion energy consumption for hovering and flying. Therefore, the total energy consumption for region $k$ is calculated using the following equation:

$$E_k(x_k, y_k, \delta_{ik_{i=1}}^N) = E_k^{trans}(x_k, y_k, \delta_{ik_{i=1}}^N) + E_k^{comp}(x_k, y_k, \delta_{ik_{i=1}}^N) + E_k^{fly}(x_k, y_k)$$
$$+ E_k^h(x_k, y_k, \delta_{ik_{i=1}}^N) \tag{14}$$

## UAV deployment and optimization

The aim of the study is to design an intelligent UAV-based offloading strategy where the IoT devices are dynamically clustered considering the number of clusters $K$, members of the clusters, and total energy. The cluster members can be measured using the maximum

number of members in the clusters $\dot{m}(\delta_{ik})$ and the sum of the distances between the members and their cluster centers $\dot{D}(\delta_{ik})$ according to the following Equations:

$$\dot{m}(K, \delta_{ik}{}_{i=1}^{N}) = \max_{k=1}^{K}\left(\sum_{i=1}^{N}\delta_{ik}\right) \tag{15}$$

$$\dot{D}(K, \delta_{ik}{}_{i=1}^{N}) = \sum_{k=1}^{K}\sum_{i=1}^{N}\delta_{ik}dist_{ik} \tag{16}$$

The total energy $E$ over $K$ clusters are calculated using the following Equation:

$$E(K, \delta_{ik}{}_{i=1}^{N}) = \sum_{k=1}^{K}E_k(x_k, y_k, \delta_{ik}{}_{i=1}^{N}) \tag{17}$$

The optimization problem of dynamically determining the optimal number of clusters, specifying the member tasks of each cluster, is modeled as mixed-integer, nonlinear constraint optimization and is formulated as follows:

**Minimize:** 
$$F = \frac{K\,E(K, \delta_{ik}{}_{i=1}^{N})}{\dot{m}(K, \delta_{ik}{}_{i=1}^{N})\,\dot{D}(K, \delta_{ik}{}_{i=1}^{N})} \tag{18}$$

**Minimize:** 
$$F = \frac{K\sum_{k=1}^{K}E_k(x_k, y_k, \delta_{ik}{}_{i=1}^{N})}{\max_{k=1}^{K}(\sum_{i=1}^{N}\delta_{ik})\sum_{k=1}^{K}\sum_{i=1}^{N}\delta_{ik}dist_{ik}} \tag{19}$$

Subject to the following constraints:

$$C_1 = R_c^2 - (dist_{ik}^2 + H^2) \le 0, \forall i = \{1, ..., N\}, k = \{1, ..., K\} \tag{20}$$

$$C_2 = \sum_{k=1}^{K}T_k(x_k, y_k, \delta_{ik}{}_{i=1}^{N}) - T_u < 0 \tag{21}$$

$$C_3 = \sum_{k=1}^{K}E_k(x_k, y_k, \delta_{ik}{}_{i=1}^{N}) - E_u < 0 \tag{22}$$

$$C_4 = \sum_{k=1}^{K}\delta_{ik} = 1 \,\forall\, i = \{1, ..., N\} \tag{23}$$

$C_1$ ensures that the determined cluster members fall within the communication radius of the cluster. Constraint $C_2$ states that the overall time required for the transmission and computations over all regions is less than the maximum flaying time of the UAV, $T_u$. Constraint $C_3$ states that the overall energy required for the UAV over all regions, including communication and computation as well as hovering, is less than the maximum energy of the UAV, $E_u$. $C_4$ states that the offloading decision of each task is allocated to

only one offloading region. To address the constraints, a penalty value is added to the optimization function for each violated constraint using the following Equation:

$$\dot{F} = F + \lambda \sum_{c_k} C_k^2 \tag{24}$$

where $\lambda$ is a penalty value. $C_k^2$ is the value of the violated constraint $k$ squared.

# PROPOSED META-HEURISTICS

UAV energy, deployment, UAV trajectory, and task delays are key challenges in a UAV-based MEC architecture. Obtaining the optimal solution for this problem using greedy search and exhaustive search optimization requires very high complexity algorithm with extremely high execution time. Also, the optimization problems involves many discrete and continuous decision parameters, including number, location of deployment, the offloading decision in each deployment, and the shortest trajectory path. Further, deep learning approaches require training data, which may not be feasible in a mobile and stochastic environment. Therefore, the optimization problem is divided into two subproblem in order to find a near optimal solution in real-time for such complex (NP)-hard optimization problem in an acceptable complexity and time. The first subproblem is the clustering the IoT devices into groups where the UAV hovers over each group for processing the offloaded tasks of each group members. In this section, we investigate the use of evolutionary meta-heuristics, including differential evolution (DE), particle swarm optimization (PSO), and genetic algorithms (GAs), to obtain a near-optimal solution for the clustering problem considering energy and time. Also, the second subproblem is the determination of the shortest UAV trajectory over the clusters head considering the time and energy of the UAV flying. Also, in this section, we investigate the use of the ACO optimization algorithm to identify the shorest trajectory that satisfy the time and energy constrains. Evolutionary and nature-inspire optimization meta-heuristics, such as DE, PSO, GA, and ACO, have the potential to obtain a near-optimal solution in an acceptable time (*Islambouli & Sharafeddine, 2019*). The following subsections describe the use of different meta-heuristics, namely DE, PSO, and ACO, to optimize the UAV-based MEC offloading system.

## DE meta-heuristic

DE is a population-based stochastic search process. DE has a few control parameters and strong search capabilities compared to most search meta-heuristics and is able to solve complex, nonlinear optimization functions (*Deng et al., 2021*). The DE simulates biological evolution in nature using mutation, crossover, and selection operations. The process iterates for a fixed number of iterations, and the population is continuously updated. The process starts with initialization of the population, where $Np$ individuals, $X_i$, $i \in [1\ Np]$, are randomly initialized. Each individual represents a possible solution to the underlying problem. The DE mutation step generates new candidate individuals $V_i$ to provide

diverse mutants for better exploration of the search space. The most common mutation strategies are:

**DE/rand/1**     $V = X_{r1} + F.(X_{r2} - X_{r3})$                                    (25)

**DE/best/1**     $V = X_{best} + F.(X_{r1} - X_{r2})$                                  (26)

where $r1$, $r2$, and $r3$ are randomly generated integers numbers $\in [1\ Np]$. $F$ is a positive factor to scale difference individuals. $X_{best}$ is the best individual with the best fitness value in the population.

In the crossover operator, the target individual $X_i$ is combined with mutant individual $V_i$, which resulted from the mutation operation, to produce trial individual $U_i$ according to probability $cr$ using the following Equation:

$$U_{ij} = \begin{cases} V_{ij}, & \text{if } rand \leq cr \text{ or } j = jrand \\ X_{ij}, & \text{otherwise} \end{cases}$$                                    (27)

where $jrand$ is a random integer from the dimension of the search space.

In the selection process, a greedy selection is applied to the trial and corresponding individual from the previous generation. The objective function value of the trial individual $f(U_i)$ is compared with the objective function value of the corresponding target vector $f(X_i)$, and the individual with the least function value will survive to the next generation.

$$X_i = \begin{cases} U_i, & \text{if } f(U_i) < f(X_i) \\ X_i, & \text{otherwise} \end{cases}$$                                    (28)

### Proposed DDE algorithm

The first step in the proposed algorithm is the encoding of the individual solution. Each individual $X_i$ is composed of $N$ genes, representing the IoT devices. Each gene will have an ID of the device, which will be the center of the cluster. The center of each cluster represents the cluster head where certain IoT devices belongs to, in which the members only offload their computation tasks once the UAV hoovers over that cluster head. For example, $X_1 = \{2, 2, 2, 4, 4, 4\}$ represents an individual solution for six IoT devices, and there are two clusters. The first cluster contains three devices 1, 2, and 3, and the second cluster contains three devices 4, 5, and 6. Devices $\{2, 4\}$ are the centers of the clusters. This section shows that the cluster heads and their members are optimized by applying the proposed DDE algorithm using the objective function shown in Eq. (24) which considers the total time, energy, number of clusters, cluster members, and the distances between the members in the clusters.

The DE algorithm is designed to solve continuous optimization problems. Therefore, the algorithm must be converted to output a discrete value. A discrete differential evolution (DDE) is designed using mutation and crossover operators. Algorithm 1 shows the detailed steps of the proposed DDE algorithm. The first step of the algorithm is the initialization of the parameters followed by the random initialization of the population

---

**Algorithm 1** Proposed DDE algorithm.

**Result:** *bestFitness*, $X_{best}$

Initialize $N$, $N_p$, $N_{iter}$, $cr$, and $F$

Random initialize $X_i$ with random clustering IDs

Calculate the fitness of each possible solution $X_i$ using an optimization function with penalty, Eq. (24)

Set *bestFitness* = $min(X)$

Set $X_{best}$ for the individual with the minimum fitness

**for** *each iter in* $N_{iter}$ **do**

    **for** *each* $X_i$ *in* $N_p$ **do**

        Apply the mutation operation, and calculate the mutated individual $V_i(t)$ using Eq. (29)

        Apply the crossover operation, and calculate the target individual $U_i(t)$ using Eq. (27)

        Apply the greedy selection, calculate the new population $X_i(t + 1)$ using Eq. (28)

    **end**

    Calculate the fitness of each possible solution in the new population

    **if** *bestFitness > best fitness of the new population* **then**

        update $X_{best}$ and *bestFitness*

    **end**

**end**

---

individuals. After that, the mutation step using Eq. (25), is changed to the following Equation:

$$V = f_3(X_{best}, f_2(F \oplus f_1(X), X_{r1})) \tag{29}$$

where $f_1$ represents a mutation operation on individual $X$ with probability $F$. For each gene in the individual, the random number $r \in [0 \ 1]$ is generated, and the gene is perturbed with a random value while $r < F$. A single point crossover operator, $f_2$, is applied to the resulting perturbed individual with a random individual from the population. A uniform crossover, $f_3$, is applied to the result of the single point crossover with the best solution found, $X_{best}$. A greedy selection is applied to the crossover operators $f_2$ and $f_3$. The last step involves the crossover and the greedy selection operations described using Eqs. (27) and (28) are applied.

## PSO meta-heuristic

The PSO is a stochastic, population-based, search meta-heuristic that simulates the social behavior of a swarm of birds searching for food. Each particle, an individual bird in the swarm, searches for food in the search space to identify the best food source based on the particle position, particle best found solution, and swarm best found global solution (*Zakaryia, Ahmed & Hussein, 2020*). Each individual particle represents a possible solution in the search space of the problem under observation. In the iterative search process of PSO, each individual particle $X_i$ updates its value $X_i(t + 1)$ by updating its corresponding velocity $V_i(t + 1)$ using the following Eqs. (30) and (31):

$$V_i(t+1) = \omega V_i(t) + C_1 \alpha (Xl_i - X_i) + C_2 \beta (Xg - X_i(t)) \tag{30}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{31}$$

where $V_i$ and $X_i$ are the velocity position and particle position, respectively, of individual $i$, and $t$ is the iteration number. $Xl_i$ is the best position of individual $X_i$, and $Xg$ is the global best position for all individuals. $\omega$ is the inertia weight. $C_1$ and $C_2$ are the cognitive coefficient and social coefficient. $\alpha$ and $\beta$ are random numbers $\in N[0\ 1]$. PSO is an effective solution for optimization problems in the continuous domain, and a discrete version of the PSO algorithm must be adopted to suit the problem under study (*Zakaryia, Ahmed & Hussein, 2020*). A new Discrete PSO (DPSO) is designed and presented in the following subsection.

## DPSO algorithm

In the DPSO, each individual is updated using mutation and crossover operators similar to the method used in DDE. The same encoding described in the DDE algorithm is employed. The particle position is updated using mutation, and a crossover operator is proposed using the following Equation:

$$X_i = C_2 \oplus f_3(C_1 \oplus f_2(\omega \oplus f_1(X_i), Xl_i), X_g) \tag{32}$$

where $\omega \oplus f_1(X_i)$ represents a mutation operation applied on individual $X_i$ with probability $w$. $f_2$ is a single point crossover operator, and $f_3$ is a uniform crossover operator. $C_1$ and $C_2$ serve as probabilities for the single point and uniform crossover operators.

Algorithm 2 shows the steps of the proposed DPSO algorithm. The algorithm starts by initializing the parameters, including the number of devices $N$, population size $N_p$, number of iterations $N_{iter}$, and mutation and crossover probabilities $\omega$, $C_1$, and $C_2$. The second step involves initializing the particles with random numbers. In the third step, an evaluation of the population with the objective function is performed using Eq. (24). Steps 5, 6, the local best of each particle and the global best of all particles, are updated. Steps 7–14 are iterated for $N_{iter}$ iterations. In each iteration, each particle is updated with mutation and crossover operators using Eq. (32), and a greedy selection is applied to the new individuals. The fitness of each particle in the new population is updated. The local best and global best are updated.

## UAV shortest trajectory using ACO

The solution of the proposed DDE algorithm identifies a set of clusters with their members. The following algorithm optimizes the shortest path among the cluster centers for the UAV to traverse using ACO. Compared to the classical algorithms of path planning, the ACO has the capability to deal with complex, dynamically changing environments, and several research studies have shown that ACO can obtain a near optimal path solution for the shortest path problem in an acceptable computational time for small and large-scale networks (*Ouyang et al., 2021*). Also, the ACO meta-heuristic has the advantage that it can escape the local optima and finding the global optima (*Puris, Bello & Herrera, 2010*).

---

**Algorithm 2** Proposed DPSO algorithm.

**Result:** *bestFitness, $X_{best}$*

Initialize $N$, $N_p$, $N_{iter}$, $\omega$, $C_1$ and $C_2$

Randomly initialize $X_i$ and $V_i$ with random clustering IDs

Calculate the fitness of each possible solution $X_i$ using an optimization function with penalty, Eq. (24)

Update the best position $Xl_i$ for each individual

Update the global position $X_g$ for all individuals

Set $X_g = min(Xl)$

**for** *each iter in $N_{iter}$* **do**

    **for** *each $X_i$ in $N_p$* **do**

        Apply mutation and crossover operations using Eq. (32)

    **end**

    Calculate the fitness of each possible solution in the new population

    Update $Xl_i$ for each individual

    Update $X_g = min(Xl)$

**end**

---

ACO is a meta-heuristic search algorithm that is based on the behavior of a group of ants in the process of searching for food. The behavior of the group optimizes an objective through a feedback strategy that ants use to identify the optimal path between the colony and the food source. Ants start leaving the colony and select random path searching of food sources. Ants communicate with each other by releasing a chemical, referred to as a pheromone, with a quantity proportional to the quantity and quality of the discovered food. This pheromone is considered an evaluation for the optimization objective. Eventually, all ants discover the shortest path by following the path corresponding to the highest pheromone concentration. The ACO meta-heuristic is an efficient search algorithm for NP-hard problems, including traveling salesman, job shop scheduling, and scheduling in mobile edge computing (*Hussein, Mousa & Alqarni, 2019*).

Given the set of points $K$ that represent the cluster head locations. The goal is to identify the shortest path among the points for the UAV to traverse for hovering to process the offloaded tasks. The length of the path from the first cluster head to the last cluster head is considered because the flying time and the flying energy are completely dependent on the length of the trajectory path of the UAV. The following two conditions must be satisfied on the the determined UAV trajectory path:

$$C_1 = T + T^{fly} \leq T_u \tag{33}$$
$$C_2 = E + E^{fly} \leq E_u \tag{34}$$

where $C_1$ ensures that the total time including communication, computation, and flying, $T^{fly}$, is less than the total flying time of the UAV $T_u$. Also, constraint $C_2$ states that the overall energy including computation, communication, hovering and flying is less than the

maximum energy of the UAV, $E_u$. $T^{fly}$ and $E^{fly}$ are calculated using the following two Equations:

$$T^{fly} = \frac{\sum_{i=1,j=i+1}^{K-1} ||u_i - u_j||^2}{\mu} \tag{35}$$

$$E^{fly} = \Gamma\, T^{fly} \tag{36}$$

where $\Gamma$ and $\mu$ are the flying energy factor and the UAV speed between the clusters head, and $u_i$ is the position of the cluster head $i$.

Initially, all ants are randomly placed on cluster heads. The algorithm iterates for a fixed number of iterations $N_{iter}$. During each iteration, each ant $m$ selects the next cluster head $j$ from cluster $i$ with probability $P_{ij}^m(t)$, calculated using Eq. (37).

$$P_{ij}^k(t) = \frac{(\tau_{ij}(t))^\alpha (\eta_{ij}(t))^\beta}{\sum_j (\tau_{ij}(t))^\alpha (\eta_{ij}(t))^\beta} \tag{37}$$

where $t$ is an iteration number, and $\alpha$ is a heuristic variable that controls the effect of the pheromone quantity and $\beta$ is a heuristic parameter that specifies the quality of the node selection. $\eta_{ij}(t)$ is a heuristic function that represents the quality of the next node selection; it is calculated using Eq. (38).

$$\eta_{ij}(t) = \frac{Q}{d(i,j)} \tag{38}$$

where $Q$ is a constant and $d(i, j)$ is the Euclidean distance between the points $i$ and $j$. $\tau_{ij}^m(t)$ represents the pheromone trail quantity for the selected path for an ant $m$ in iteration $t$, calculated using Eq. (39).

$$\tau_{ij}^m(t+1) = (1-\rho)\tau_{ij}^m(t) + \rho\Delta\tau_{ij}^m(t) \tag{39}$$

where $\Delta\tau_{ij}^m(t) = 1/L^m$ and $\rho$ is a constant that represents the rate of pheromone evaporation at each step. $L^m$ is the length of the trajectory path determined by ant $m$.

Once all ants have completed their path finding, *i.e.*, a complete iteration has been performed, the pheromone trail is updated globally. The global pheromone update is calculated using Eq. (40).

$$\tau_{ij}^m(t+1) = (1-\rho_g)\tau_{ij}^m(t) + \rho_g\Delta\tau_{ij}^m(t) \tag{40}$$

where $\Delta\tau_{ij}^m(t) = 1/L_{best}$, $L_{best}$ is the best shortest path discovered, and $\rho_g$ is the global evaporation rate.

Algorithm 3 shows the steps of the proposed ACO-based UAV shortest path among a set of cluster heads. The proposed algorithm starts by initializing the parameter settings $\alpha$ and $\beta$; the local pheromone concentration $Q$; the local pheromone decay parameter $\rho$; the global pheromone decay parameter $\rho_g$; the number of ants $N_{ants}$; the number of iterations $N_{iter}$; and the number of cluster heads points $N_k$. Step 2 initializes the pheromone matrix $\tau_{ij}^m(0) = 1/d(i,j)$ for each ant $m$. During each iteration, for each ant, $ant_m$

---

**Algorithm 3** The ACO-based UAV trajectory planning algorithm.

---

**Result:** Shortest path for the UAV to traverse.

Initialize the parameters $\alpha$, $\beta$, $Q$, $\rho$, $\rho_g$, $N_{ants}$, $N_{iter}$, and $N_k$

Create an initial population of ants

Initialize the pheromone matrix $\tau_{ij}^m(0) = 1/d(i,j)$

**While** $iter \leq N_{iter}$ **do**

    Randomly place all ants at the starting nodes

    **for** each $ant_m$ **do**

        **while** $ant_m$ has not finished its journey **do**

            **for** each $j$ in $N_k \notin Journey^m$ **do**

                `//ADD node j to the current journey of the` $ant_m$ `solution`

                Calculate $\eta_{ij}^m(t)$ using Eq. (38)

                Calculate $P_{ij}^m(t)$ using Eq. (37)

                $ant_m$ chooses the cluster head $j$ using the roulette method

            **end**

        **end**

        Update $\tau_{ij}^m(t+1)$ using Eq. (39)

    **end**

    Compare all ant solutions with the previous best solution and update the best solution *Journeybest*

    **if** the current solution is the best **then**

        Update $\tau_{ij}^m(t+1)$ using Eq. (40)

**end**

**end**

---

searches for a complete path journey. All ants are randomly placed on initial points. The selection of the next point $j$ is calculated with probability $P_{ij}^m(t)$ using Eq. (37). The selection of the next point is performed using the roulette wheel selection method (*Holland, 1992*). When an ant finishes a complete journey, the local pheromone trail matrix is updated using Eq. (39). After each iteration, the ACO-based algorithm updates the global pheromone trail matrix $\tau_{ij}^m(t)$ using Eq. (40). The algorithm iterates until the maximum number of iterations is reached.

## EXPERIMENTAL RESULTS

This section provides the simulation results that verify the performance of the proposed UAV-based offloading systems using the proposed meta-heuristics. The IoT devices are randomly deployed in a 200 $m \times$ 200 $m$ area with the simulation parameters that are established according to the values described in Table 2. Different possible values of the meta-heuristics parameters are tested in the experiments, and the best value that obtained the best results are reported. However, there are many methods for tuning hyperparameters of the meta-heuristics, including grid search, artificial neural network,

**Table 2 Parameter settings for the evaluation experiments.**

| Parameter | Description | Value |
|-----------|-------------|-------|
| $B$ | Bandwidth | 40 *MHz* |
| $h_i$ | Channel gain | −30 dB |
| $N_0$ | Noise power | $10^{-9}$ *W* |
| $\kappa_1$ | UAV energy communication coefficient | $10^{-18}$ |
| $\kappa_2$ | UAV energy computation coefficient | $10^{-26}$ |
| $\gamma$ | UAV energy hoovering coefficient | $10^{-10}$ *W* |
| $\Gamma$ | UAV energy flying coefficient | $10^{-8}$ *W* |
| $P_{ik}$ | Maximum transmission power | 0.4 W |
| $\mu$ | Maximum flying speed of the UAV | 20 *m/s* |
| $H$ | Flying altitude of the UAV | 50 *m* |
| $t^{data}$ | Task data size | 200 *KB*−3 *MB* |
| $t^c$ | Required task computing cycles | $6 \times 10^9$−$9 \times 10^{10}$ |
| $f^u$ | CPU-cycle frequency of the UAV | 300 *MHz* |
| $E_u$ | Battery storage capacity of the UAV | $5 \times 10^5 J$ |
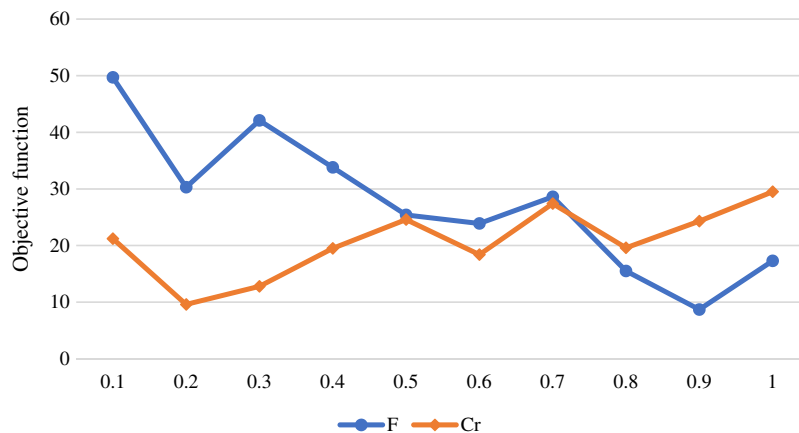
**Figure 1 The value of the objective function for different values of the DDE algorithm parameters.**

and bayesian optimization across continuous spaces which can be considered in our future works. The proposed DDE algorithm has two parameters *F* and *Cr*, and Fig. 1 shows the impact of the different values of the DDE algorithm parameters on the value of the objective function. The figure shows that the best values that maintain lowest objective function value are F = 0.9 and CR = 0.2. The GA has a crossover value set to 1 and a mutation probability set to 0.05. The DPSO parameters are set to $w = 0.2$, $c_1 = 0.9$, and $c_2 = 0.9$. The maximum number of iterations is set to 500. The collected results are based on applying 10 different runs, and the average is calculated.

To evaluate the convergence of the proposed meta-heuristics in terms of clustering efficiency, Fig. 2 shows the value of the objective function of the DDE, DPSO, and GA
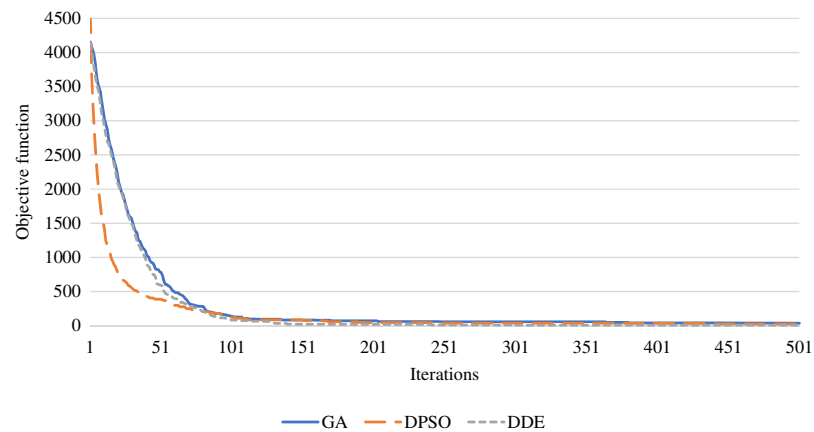
**Figure 2** The value of the objective function using the proposed meta-heuristics during the iterations.
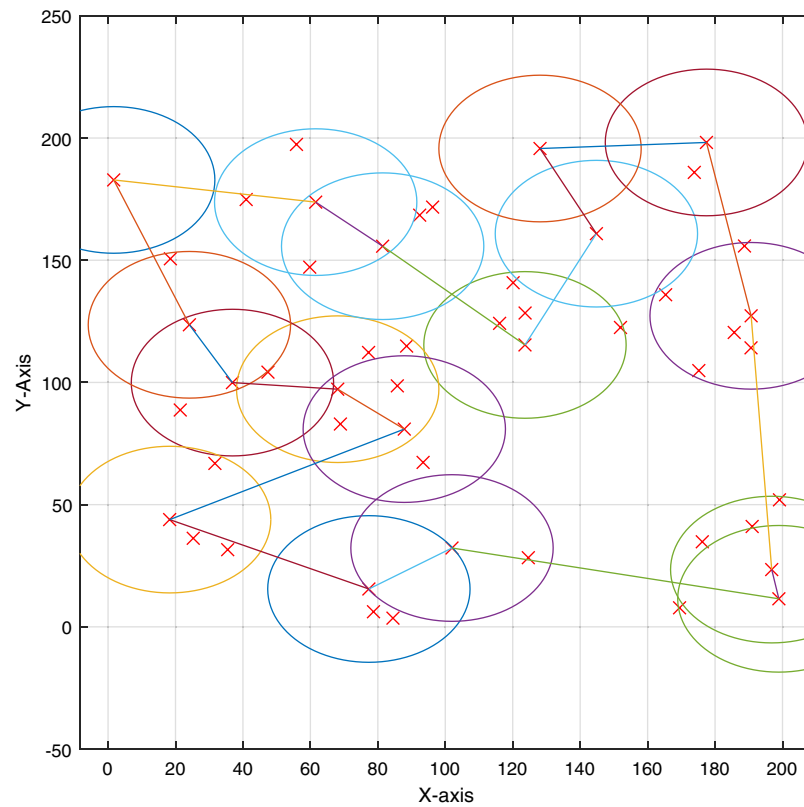
**Figure 3** Example of the clustering process using the DDE algorithm for 30 IoT devices.

meta-heuristics during the iterations of a single run. The DDE algorithm is capable of achieving the best convergence compared to DPSO and the GA within the fewer number of iterations. Also, the DPSO with with the mutation and crossover operators can achieve better performance than the GA algorithm.
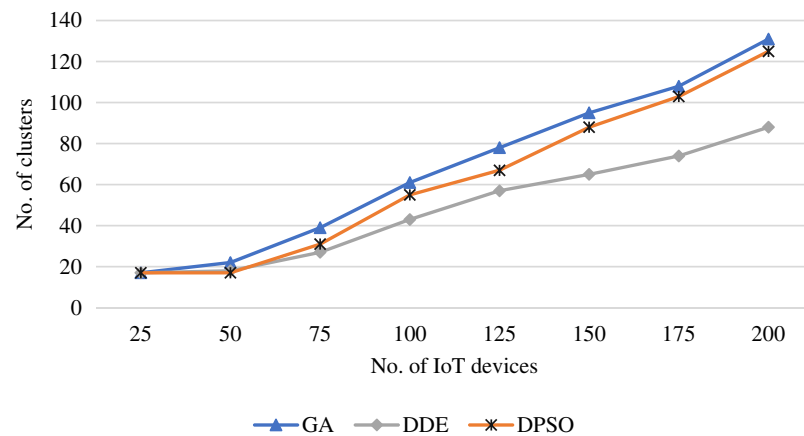
**Figure 4 Average number of clusters obtained using the proposed meta-heuristics as increasing the IoT devices.**
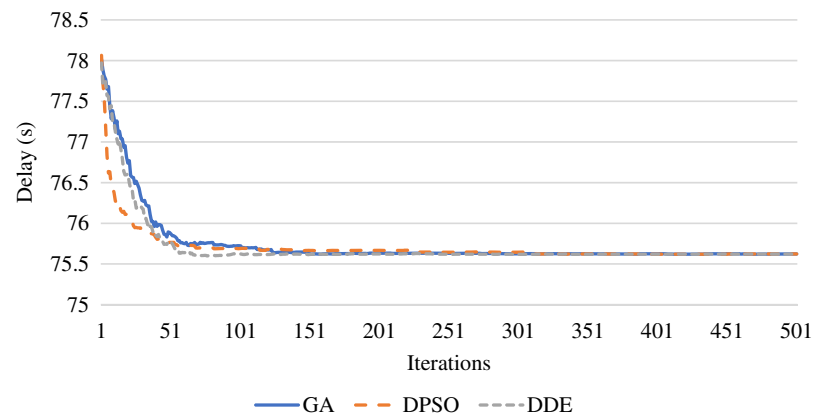
**Figure 5 Optimization of the time delay using the DDE, PSO, and GA algorithms.**

Figure 3 shows the clusters resulting from the DDE algorithm for 50 IoT devices. The algorithm successfully produced 30 clusters and their members fall within the communication radius of the cluster heads. Further, the shortest path for the UAV is efficiently produced using the ACO algorithm. To evaluate the clustering process in terms of the number of clusters, Fig. 4 shows the resulting average number of clusters using the proposed meta-heuristics for different runs as increasing the number of IoT devices. The proposed DDE algorithm maintains the lowest number clusters. The DDE algorithm is able to obtain the global optimum compared to the PSO and GA algorithms, and thus, a better clustering is maintained using the DDE algorithm.

For the evaluation in terms of time delay, Fig. 5 shows the resulting delay during the iterations of the algorithms. The figure assures that the DDE algorithm is capable of identifying the lowest time delay faster than the DPSO and GA algorithms, which
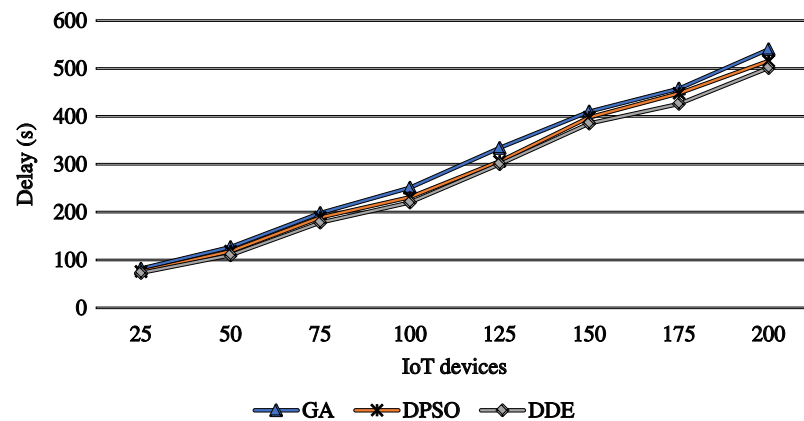
**Figure 6 Average time delay obtained using the proposed meta-heuristics as increasing the IoT devices.**
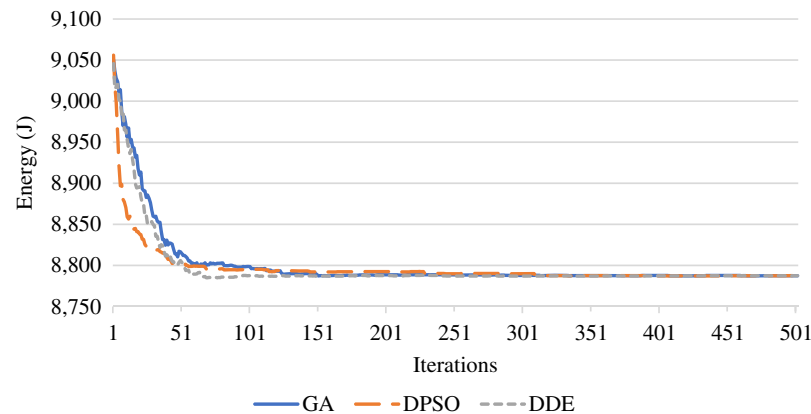
**Figure 7 Optimization of the UAV energy using the DDE, PSO, and GA algorithms.**

effectively reduces the time delay. Further, Fig. 6 presents the average total time delay with an increasing number of IoT devices. This figure shows that the proposed strategy outperforms the comparison algorithm and successfully decreases the delay with an increasing number of offloaded tasks.

In terms of energy, Fig. 7 shows the energy achieved during the iterations of a single run. The proposed meta-heuristics are capable of reducing the energy while the DDE algorithm maintains the lowest energy consumption. Further, Fig. 8 shows the average of the UAV energy as increasing the number of the IoT devices. This figure shows that as the number of IoT devices increases, the corresponding energy consumption increases. This increase is attributed to increasing the transmission and processing times, and consequently, the hovering time will increase. The DDE algorithm outperforms the PSO and GA algorithms and saves energy.
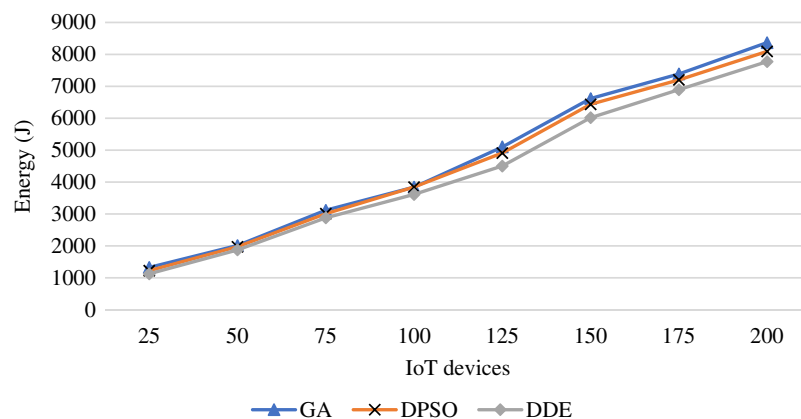
**Figure 8** Average energy achieved using the proposed meta-heuristics as increasing the IoT devices.

## CONCLUSION AND FUTURE WORK

This paper proposed a UAV-based offloading system that is aimed at minimizing the delay and the UAV energy consumption. Two different subproblems for the proposed system are investigated. The first subproblem is the clustering of the IoT devices, where the proposed offloading system partitions the ground devices into clusters, and the UAV hovers over each cluster head to process the offloaded tasks for the cluster members. The second problem is the shortest path for the UAV to traverse the cluster heads. For the first sub-problem, two different discrete meta-heuristic are proposed for the clustering process, namely, DDE and DPSO. The proposed meta-heuristics are compared with the GA. For the second subproblem, ACO is employed to identify the shortest path among the cluster heads. The experimental results show the effectiveness of the proposed offloading system using DDE and ACO in terms of delay, energy, the number of resulting clusters, and optimized trajectory of the UAV. Future work will consider the mobility of ground devices. We will investigate UAV-based, offloading MEC with multi-cooperative UAVs to study the effect of using multi-cooperative UAVs on the scope of the problem and the performance of the proposed strategy algorithm. Another interesting point for the future work is to test the proposed offloading strategy on real-time tasks where the tasks execution time must meet a specified delay tolerance in the optimization, and to do a complete analysis on different type of tasks, including compute-bound, network-bound, and real-time.

### Funding

This work was funded by the University of Jeddah, Saudi Arabia, under grant No. (UJ-20-102-DR). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

## Grant Disclosures

The following grant information was disclosed by the authors:
University of Jeddah, Saudi Arabia: UJ-20-102-DR.

## Competing Interests

The authors declare that they have no competing interests.

## Author Contributions

- Mohamed H. Mousa conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Mohamed K. Hussein conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.

## Data Availability

The code is available at GitHub: https://github.com/mkhamiss/PeerJ.git.

## REFERENCES

**Bejaoui A, Park K, Alouini M. 2020.** A QoS-oriented trajectory optimization in swarming unmanned-aerial-vehicles communications. *IEEE Wireless Communications Letters* **9(6)**:1 DOI 10.1109/LWC.2020.2970052.

**Chen J, Chen S, Luo S, Wang Q, Cao B, Li X. 2020.** An intelligent task offloading algorithm (ITOA) for UAV edge computing network. *Digital Communications and Networks* **6(4)**:433–443 DOI 10.1016/j.dcan.2020.04.008.

**Cheng N, Xu W, Shi W, Zhou Y, Lu N, Zhou H, Shen X. 2018.** Air-ground integrated mobile edge networks: architecture, challenges, and opportunities. *IEEE Communications Magazine* **56(8)**:26–32 DOI 10.1109/MCOM.2018.1701092.

**Deng W, Shang S, Cai X, Zhao H, Song Y, Xu J. 2021.** An improved differential evolution algorithm and its application in optimization problem. *Soft Computing* **25(7)**:5277–5298 DOI 10.1007/s00500-020-05527-x.

**Fu S, Tang Y, Wu Y, Zhang N, Gu H, Chen C, Liu M. 2021.** Energy-efficient UAV-enabled data collection via wireless charging: a reinforcement learning approach. *IEEE Internet of Things Journal* **8(12)**:10209–10219 DOI 10.1109/JIOT.2021.3051370.

**Fu S, Tang Y, Zhang N, Zhao L, Wu S, Jian X. 2020.** Joint unmanned aerial vehicle (UAV) deployment and power control for internet of things networks. *IEEE Transactions on Vehicular Technology* **69(4)**:4367–4378 DOI 10.1109/TVT.2020.2975031.

**Guo H, Liu J. 2020.** UAV-enhanced intelligent offloading for internet of things at the edge. *IEEE Transactions on Industrial Informatics* **16(4)**:2737–2746 DOI 10.1109/TII.2019.2954944.

**He H, Zhang S, Zeng Y, Zhang R. 2018.** Joint altitude and beamwidth optimization for UAV-enabled multiuser communications. *IEEE Communications Letters* **22(2)**:344–347 DOI 10.1109/LCOMM.2017.2772254.

**Holland JH. 1992.** *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence.* Cambridge, MA, USA: MIT Press.

**Hu Q, Cai Y, Yu G, Qin Z, Zhao M, Li GY. 2019a.** Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. *IEEE Internet of Things Journal* **6(2)**:1879–1892 DOI 10.1109/JIOT.2018.2878876.

**Hu B, Ding X, Yang F, Liu J. 2019.** A modified algorithm for computation issues in UAV-enabled wireless communications. *EURASIP Journal on Wireless Communications and Networking* **2019(1)**:269 DOI 10.1186/s13638-019-1593-z.

**Hu B, Sun Z, Hong H, Liu J. 2020.** UAV-aided networks with optimization allocation via artificial bee colony with intellective search. *EURASIP Journal on Wireless Communications and Networking* **2020(1)**:40 DOI 10.1186/s13638-020-1659-y.

**Hu X, Wong K, Yang K, Zheng Z. 2019b.** UAV-assisted relaying and edge computing: scheduling and trajectory optimization. *IEEE Transactions on Wireless Communications* **18(10)**:4738–4752 DOI 10.1109/TWC.2019.2928539.

**Hussein MK, Mousa MH. 2020.** Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access* **8**:37191–37201 DOI 10.1109/ACCESS.2020.2975741.

**Hussein MK, Mousa MH, Alqarni MA. 2019.** A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing* **8(1)**:7 DOI 10.1186/s13677-019-0131-1.

**Islambouli R, Sharafeddine S. 2019.** Optimized 3D deployment of uav-mounted cloudlets to support latency-sensitive services in iot networks. *IEEE Access* **7**:172860–172870 DOI 10.1109/ACCESS.2019.2956150.

**Lan Y, Wang X, Wang C, Wang D, Li Q. 2019.** Collaborative computation offloading and resource allocation in cache-aided hierarchical edge-cloud systems. *Electronics* **8(12)**:1430 DOI 10.3390/electronics8121430.

**Li M, Cheng N, Gao J, Wang Y, Zhao L, Shen X. 2020a.** Energy-efficient UAV-assisted mobile edge computing: resource allocation and trajectory optimization. *IEEE Transactions on Vehicular Technology* **69(3)**:3424–3438 DOI 10.1109/TVT.2020.2968343.

**Li L, Wen X, Lu Z, Jing W. 2020b.** An energy efficient design of computation offloading enabled by UAV. *Sensors* **20(12)**:3363 DOI 10.3390/s20123363.

**Mao Y, You C, Zhang J, Huang K, Letaief KB. 2017.** A survey on mobile edge computing: the communication perspective. *IEEE Communications Surveys Tutorials* **19(4)**:2322–2358 DOI 10.1109/COMST.2017.2745201.

**Mohamed N, Al-Jaroodi J, Jawhar I, Noura H, Mahmoud S. 2017.** UAVFog: a UAV-based fog computing for internet of things. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI).* Piscataway: IEEE, 1–8.

**Nguyen V, Khanh TT, Van Nam P, Thu NT, Seon Hong C, Huh E. 2020.** Towards flying mobile edge computing. In: *2020 International Conference on Information Networking (ICOIN).* 723–725.

**Ouyang Y, Liu W, Yang Q, Mao X, Li F. 2021.** Trust based task offloading scheme in UAV-enhanced edge computing network. *Peer-to-Peer Networking and Applications* **14(5)**:3268–3290 DOI 10.1007/s12083-021-01137-y.

**Puris A, Bello R, Herrera F. 2010.** Analysis of the efficacy of a two-stage methodology for ant colony optimization: case of study with TSP and QAP. *Expert Systems with Applications* **37(7)**:5443–5453 DOI 10.1016/j.eswa.2010.02.069.

**Tang Q, Chang L, Yang K, Wang K, Wang J, Sharma PK. 2020.** Task number maximization offloading strategy seamlessly adapted to uav scenario. *Computer Communications* **151(3)**:19–30 DOI 10.1016/j.comcom.2019.12.018.

**Wang R, Cao Y, Noor A, Alamoudi TA, Nour R. 2020a.** Agent-enabled task offloading in UAV-aided mobile edge computing. *Computer Communications* **149(4)**:324–331 DOI 10.1016/j.comcom.2019.10.021.

**Wang J, Jiang C, Wei Z, Pan C, Zhang H, Ren Y. 2019.** Joint uav hovering altitude and power control for space-air-ground IoT networks. *IEEE Internet of Things Journal* **6(2)**:1741–1753 DOI 10.1109/JIOT.2018.2875493.

**Wang Y, Ru ZY, Wang K, Huang PQ. 2020b.** Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing. *IEEE Transactions on Cybernetics* **50(9)**:3984–3997 DOI 10.1109/TCYB.2019.2935466.

**Wang L, Wang K, Pan C, Xu W, Aslam N, Nallanathan A. 2021.** Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing. *IEEE Transactions on Mobile Computing*. Epub ahead of print 16 February 2021 DOI 10.1109/TMC.2021.3059691.

**Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W. 2017.** A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access* **5**:6757–6779 DOI 10.1109/ACCESS.2017.2685434.

**Wu F, Yang D, Xiao L, Cuthbert L. 2019.** Energy consumption and completion time tradeoff in rotary-wing UAV enabled wpcn. *IEEE Access* **7**:79617–79635 DOI 10.1109/ACCESS.2019.2922651.

**Wu W, Zhou F, Hu RQ, Wang B. 2020.** Energy-efficient resource allocation for secure noma-enabled mobile edge computing networks. *IEEE Transactions on Communications* **68(1)**:493–505 DOI 10.1109/TCOMM.2019.2949994.

**Yang L, Yao H, Wang J, Jiang C, Benslimane A, Liu Y. 2020.** Multi-UAV-enabled load-balance mobile-edge computing for iot networks. *IEEE Internet of Things Journal* **7(8)**:6898–6908 DOI 10.1109/JIOT.2020.2971645.

**Yu Y. 2016.** Mobile edge computing towards 5G: vision, recent progress, and open challenges. *China Communications* **13(Supplement2)**:89–99 DOI 10.1109/CC.2016.7405725.

**Yu Z, Gong Y, Gong S, Guo Y. 2020.** Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet of Things Journal* **7(4)**:3147–3159 DOI 10.1109/JIOT.2020.2965898.

**Zakaryia SA, Ahmed SA, Hussein MK. 2020.** Evolutionary offloading in an edge environment. *Egyptian Informatics Journal* **22(3)**:257–267 DOI 10.1016/j.eij.2020.09.003.

**Zhan C, Hu H, Sui X, Liu Z, Niyato D. 2020.** Completion time and energy optimization in the uav-enabled mobile-edge computing system. *IEEE Internet of Things Journal* **7(8)**:7808–7822 DOI 10.1109/JIOT.2020.2993260.

**Zhang J, Chen T, Zhong S, Wang J, Zhang W, Zuo X, Maunder RG, Hanzo L. 2019.** Aeronautical ad hoc networking for the internet-above-the-clouds. *Proceedings of the IEEE* **107(5)**:868–911 DOI 10.1109/JPROC.2019.2909694.

**Zhou F, Hu RQ, Li Z, Wang Y. 2020.** Mobile edge computing in unmanned aerial vehicle networks. *IEEE Wireless Communications* **27(1)**:140–146 DOI 10.1109/MWC.001.1800594.