Joint Differential Evolution and Successive Convex Approximation in UAV-enabled Mobile Edge Computing

ZHE YU, (Graduate Student Member, IEEE), AND GUOLIANG FAN, (Senior Member, IEEE)
School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA
Corresponding author: Guoliang Fan (e-mail: guoliang.fan@okstate.edu).

Abstract UAV-enabled mobile edge computing (MEC) is a emerging technology to support resource-intensive yet delay-sensitive applications with edge clouds (ECs) deployed in the proximity to mobile users and UAVs served as computing base stations in the air. The formulated optimization problems therein are highly nonconvex and thus difficult to solve. To tackle the nonconvexity, the successive convex approximation (SCA) technique has been widely used to solve for the nonconvex optimization problems by transforming the nonconvex objective functions and constraints into suitable convex surrogates. However, the optimal solutions are based on the approximated optimization problem not the original one and they are highly dependent on the feasible solution initialization. Unlike SCA, Differential Evolution (DE) is a global optimization method that iteratively updates the best candidate solutions with respect to the predefined objective functions. DE works well especially in unconstrained optimization problems since it can freely search very large regions of possible solutions without considering the convexity of the original problem. However, when it comes to the constrained optimization problem, DE becomes inefficient to find the feasible and optimal solutions within given time limits. In view of the shortcomings incurred in both DE and SCA, we propose an innovative algorithm by jointly applying DE and SCA (DE-SCA) to solve for the nonconvex optimization problems. However, directly using full DE solutions to initialize the SCA-based algorithm will result in worse objective function values as the DE solutions are often infeasible. Therefore, we further design to screen the feasible parts from the DE solutions and utilize them to initialize the SCA-based algorithm. In experimental simulations, we consider a system of UAV-enabled MEC where IoT devices, the UAV and ECs interact with each other. The simulation results demonstrate that our proposed Screened DE-SCA algorithm largely outperforms the benchmarks including DE, SCA-based and state-of-the-art algorithms in the UAV-enabled MEC system.

Index Terms Differential evolution, mobile edge computing, successive convex approximation, unmanned aerial vehicles.

I. INTRODUCTION

UAV-enabled mobile edge computing (MEC) [1], [2] has emerged as a paradigm to support resource-intensive yet delay-sensitive applications with UAVs that are equipped with onboard communication, computing, and storage (CCS) served as airborne base stations as well as computing units and small cloud-computing platforms deployed at the mobile network edges. High altitude of UAVs can help establish better communication links by avoiding potential obstacles on the ground but the UAVs are also constrained by the size, weight, and power (SWAP). It is essential to jointly manage the computation and communication resource allocation, energy-efficiency of UAVs and system delay experienced by mobile users in the UAV-enabled MEC systems and thus they are often formulated as optimization problems. However, the formulated problems are highly nonconvex and thus difficult to solve. To tackle the nonconvexity, the successive convex approximation (SCA) [3] technique has been extensively utilized to obtain the sub-optimal solutions.

SCA can be applied to solve a series of nonconvex optimization problems by converting nonconvex objective functions and constraints into qualified convex forms while the local first-order behavior of the original nonconvex problem is preserved [4], [5]. Then, we iteratively compute the local optimum of the resulting convex problem by updating the initial feasible solutions until a stationary solution of the
original nonconvex problem is found. The reformulated convex optimization problems can be effectively solved by optimization software such as Gurobi and CVX so the feasibility of solutions can be largely guaranteed. However, there are two potential issues raised while applying the SCA to solve for nonconvex optimization problems. On one hand, SCA-based algorithms are highly dependent on the feasible solution initialization such that random initialization can usually result in local optimum. On the other hand, the found optimal solutions are only optimal with respect to the reformulated convex problem but there lacks evidence to assure that they are also optimal to the original nonconvex problem.

Meanwhile, Differential Evolution (DE) [6] has been widely applied in solving UAV-enabled MEC problems benefiting from its ability to search large space of candidate solutions imposed on the original problems. Same as other evolutionary algorithms (EAs) [7], DE was initially developed for solving unconstrained optimization problems. In such problems, the only constraints are the bounds of each decision variable, which jointly serve as the searching space for DE to work with. The potential infeasibility issue can be easily eliminated by cropping the values of decision variables in the range of their own lower bounds and upper bounds. DE can actually reduce the computational complexity of improving the candidate solutions by mimicking the process of natural selection rather than purely relying on brute-force searching. In this process, DE generally consists of four stages of initialization, mutation, crossover and selection where the differential scheme in the mutation stage is the key to differentiate DE from other EAs [8].

However, the infeasibility issue is almost inevitable when we extend DE to solve for constrained optimization problems (COPs) [9]. The equality constraints therein are inherently hard to be satisfied as they will extremely shrink the feasible region compared to the one with respect to the same constraints expressed as inequality. In EAs, the equality constraints can always be transformed into corresponding inequality counterparts with designated tolerance degree whose value will largely affect the convergence rate of the optimization algorithms. However, infeasible solutions still persist in DE since it only approximately handles the feasibility of generations during the selection stage without forcing them to satisfy all the constraints early in the initialization stage. On the contrary, if all individuals are initialized strictly by meeting all the constraints, DE can easily be trapped in the initialize-check feasibility-reinitialize loop forever as the population size is usually large in solving real problems. In other words, a tradeoff exists between feasibility and convergence when we apply DE to solve for large-scale COPs. Specifically, the pros and cons of DE and SCA-based algorithms are summarized in Table 1.

In this article, we propose an novel algorithm that jointly considers DE and SCA to overcome the drawbacks of solely applying DE or SCA to solve for nonconvex optimization problems. In this way, we not only preserve the originality of the optimization problem since DE can be directly used to solve the original problem without further reformulations but also assure the feasibility of the found solutions since the reformulated convex optimization problem by means of SCA can be effectively solved. Specifically, the proposed algorithm can be described by three sections: DE stages, screened DE solutions and SCA stages.

In DE stages, the original nonconvex problem is iteratively solved by the classic DE algorithm without the need of further problem approximation. As previously discussed, the DE solutions of a constrained optimization problem with large numbers of constraints are generally infeasible and thus we need to carefully analyze the best solutions found by applying DE algorithm. In other words, we need to figure out how to initialize the subsequent SCA-based algorithm using the infeasible solutions found on the original problem. Then, we propose to screen the feasible parts from the best DE solutions and use them to initialize the SCA-based algorithm as it requires that the initialized solutions must all be feasible. In SCA stages, we will need to transform the original problem into its convex approximated problem before we can apply SCA-based algorithm to solve. Then, we initialize the SCA-based algorithm with the screened feasible parts of the best DE solutions along with the randomly generated feasible solutions for those that are infeasible parts of the best DE solutions. After iteratively solving for the approximated problem with SCA-based algorithm, we can obtain the best SCA solutions and then plug them into the objective function of the original problem to find the optimal objective function value.

To the best of our knowledge, this is the first work to jointly consider DE and SCA algorithms to solve for nonconvex optimization problems. We further propose to screen the feasible parts from the best DE solutions to initialize the SCA-based algorithm. Our proposed method can be applied to cope with the nonconvex problem that consists of an objective function and constraints with certain structures (e.g., smoothness) that can be further approximated into a suitable convex form [4], [5]. To validate the effectiveness of our proposed method, we investigate the nonconvex optimization problem formulated in the UAV-enabled MEC system that consists of a UAV, multiple IoT devices and edge clouds (ECs) on the ground.

The main contributions of this article are summarized as follows.

1) We propose an innovative method to jointly consider DE and SCA in view of their own advantages and dis-

| Algorithms | Pros                                                                 | Cons                                      |
|------------|---------------------------------------------------------------------|-------------------------------------------|
| DE         | Searching large space of candidate solutions                       | Working on original problem without reformulations |
|            |                                                                     | Feasibility not guaranteed in COPs         |
|            |                                                                     | Slow convergence in COPs                  |
| SCA-based  | Solving nonconvex optimization problem via convex reformulations    | Subject to feasible solution initialization |

TABLE 1. Pros and Cons for DE and SCA-based Algorithms
advantages when solely applied to solve for nonconvex optimization problems.

2) To better combine with the SCA-based algorithm, we further propose to only select the feasible parts of the best DE solution of the original problem. After this screening procedure, the feasible parts will be used to initialize the SCA-based algorithm.

3) We conduct extensive simulations to evaluate the performance of our proposed Screened DE-SCA method in a UAV-enabled MEC system. Numerical experiments demonstrate that our proposed Screened DE-SCA method largely outperforms baseline methods.

The remainder of this article is organized as follows. Related work is reviewed in Section II. In Section III, we present preliminaries on constrained optimization problems, Differential Evolution and successive convex approximation. In Section IV, we describe the concrete framework and algorithm of our proposed method. In Section V, we introduce the system model of a UAV-enabled MEC system and then formulate the optimal IoT task offloading processes as a nonconvex optimization problem. The numerical experiment results based on real-world traces are analyzed in Section VI. Finally, the conclusion is summarized in Section VII.

II. RELATED WORK

In this section, we review the prior works most relevant to our paper from three aspects: 1) DE methods in UAV-enabled MEC; 2) SCA-based methods in UAV-enabled MEC and 3) alternative methods in UAV-enabled MEC.

A. DE METHODS IN UAV-ENABLED MEC

Benefiting from its ability to search large space of candidate solutions imposed on the original problems, the DE methods have been utilized to explore the best solutions of the optimization problems formulated in UAV-enabled MEC systems.

1) Auxiliary DE Methods

On one hand, DE method can be used to generate intermediate results in joint algorithms in [10]–[13]. In [10], a discrete differential evolution (DDE) algorithm along with ant colony optimization (ACO) algorithm are put forward to jointly optimize the clustering of IoT devices and UAV trajectory in a UAV-enabled MEC system. In [11], an evolutionary trajectory planning algorithm (ETPA) which adopts a DE clustering method is proposed to jointly optimize the overall system energy consumption as well as the path planning of UAVs in a multi-UAV-assisted MEC system. In [12], a multi-objective evolutionary algorithm which adopts DE to update solutions along with the deep deterministic gradient algorithm is designed to maximize the sum computation rate at all IoT devices while satisfying the energy harvesting constraints and coverage in UAV-aided wireless powered MEC networks. In [13], a trajectory planning algorithm (TPA) that adopts a DE algorithm with variable population sizes is put forward to optimize the system energy consumption via planning the UAV trajectories in a multi-UAV-associated MEC system.

2) Direct DE Methods

On the other hand, DE method can be directly applied to generate the best solutions in [14]–[17]. In [14], a novel DE algorithm with variable population size based on a mutation strategy pool initialized by K-Means is developed to minimize the energy consumption in a UAV-assisted edge data collection system. In [15], a DE-based mechanism is designed to jointly optimize the load-balancing and latency-aware task scheduling incurred in a multi-UAV-enabled MEC system. In [16], a differential evolution algorithm with a variable population size (DEVIPS) is developed to minimize the system energy consumption by optimizing the UAV deployment in a UAV-assisted IoT data collection system. In [17], a DE algorithm with an elimination operator is proposed to jointly optimize the deployment of UAVs and task scheduling for all mobile users in a multi-UAV-enabled MEC system.

However, all of these studies can only achieve near-optimal solutions due to the inherent inefficiency of DE methods to handle constraints when we extend them to solve for COPs.

B. SCA-BASED METHODS IN UAV-ENABLED MEC

Due to the ability to approximately transform the nonconvex optimization problem formulated in UAV-enabled MEC systems into a solvable convex form, the SCA-based methods have been widely exploited in the literature.

1) Auxiliary SCA-based Methods

On one hand, SCA technique can be combined with other algorithms to find solutions in [18]–[22]. In [18], the Branch and Bound (BnB) method and SCA technique are exploited to jointly optimize the service placement, task scheduling and UAV trajectory subproblems in a UAV-enabled MEC system. In [19], the Dinkelbach’s method, Lagrange duality and SCA technique are combined to jointly maximize the weighted computation efficiency subject to the constraints on resource allocation, minimum computation and UAV’s mobility in the UAV-assisted MEC networks. In [20], the SCA and block coordinate descent (BCD) algorithms are utilized to maximize the minimum secure calculation capacity in order to improve the security of communications in dual UAV MEC systems. In [21], a Dinkelbach method adopting simulated annealing and SCA is proposed to jointly optimize the gateway selection and resource allocation involved with space-air-ground IoT networks. In [22], SCA technique and Lagrangian duality method are jointly applied to minimize the total energy consumption by optimizing the computation bits allocation, time slot scheduling, transmit power allocation, and UAV trajectory in a UAV-assisted MEC system.

2) Direct SCA-based Methods

On the other hand, SCA technique can be directly used to develop SCA-based algorithms in [23]–[25]. In [23], SCA-
based algorithms are designed to jointly optimize the completion time and energy consumption of UAV as well as its trajectory in the UAV-enabled MEC system. In [24], a SCA-based algorithm is developed to jointly optimize the UAV trajectory subject to the energy harvesting causality and user scheduling constraints in UAV-enabled wireless powered communication networks. In [25], a SCA-based algorithm is presented to jointly optimize the bit allocation for communication and computation as well as the UAV trajectory in the MEC system via a UAV-mounted cloudlet.

However, the optimal solutions of these works are highly dependent on the feasible solution initialization which gives guidance on how the candidate solutions are iteratively searched until convergence. To overcome the drawbacks of solely applying DE or SCA-based methods to solve non-convex optimization problems formulated in UAV-enabled MEC systems, we propose to jointly exploit DE and SCA such that better local optimal solutions of the original problem can be obtained.

### C. ALTERNATIVE METHODS IN UAV-ENABLED MEC

Apart from DE methods and SCA-based methods, there are ample studies using alternative methods to solve related problems formulated in UAV-enabled MEC.

#### 1) Optimization Methods

Main problems that are studied in UAV-enabled MEC include task scheduling, computation offloading, user association, resource allocation and trajectory planning. There are many optimization methods to tackle these problems. To name a few, in [26], the UMEC method combing the proposed RTSA and submodularity is designed to effectively solve a mixed-integer nonlinear programming problem that is formulated to model the task selection and scheduling for reconnaissance with time-varying priorities conditions in UAV-enabled MEC. In [27], an algorithm combining the alternative optimization and successive convex programming is developed to solve a nonconvex problem aiming to maximize the uplink common throughput among all ground users over a finite UAV’s flight period in UAV-enabled wireless powered communication network. In [28], a learning-based cooperative particle swarm optimization algorithm with a Markov random field-based decomposition strategy is put forward to search for the optimal UAV resource allocation strategy for industrial IoT in UAV-enabled MEC.

#### 2) Game-theoretic Methods

There are many papers aiming to investigate games that are formulated in UAV-enabled MEC by viewing the mobile users, base stations and edge/cloud infrastructures as autonomous agents that each can have their own utility functions. To name a few, in [29], a selfish game is formulated to model the task offloading and resource competition problem in UAV-assisted multiaccess edge computing system. The Nash equilibrium is proved to be existent and a game-theoretic scheme is proposed to find the optimal solution. In [30], a two-level game model including cooperative game in the upper level and noncooperative subgames in the lower level is formulated to model the payoff maximization problem raised in UAV-enabled MEC with multiple service providers. The mixed-strategy Nash equilibrium is found by combining coalition formation with reinforcement learning. In [31], a dynamic evolutionary game is formulated to study the access competition among groups of UAVs and it is solved by an evolutionary equilibrium. Also, a noncooperative game is formulated to study the bandwidth that is allocated by base stations to the UAVs and it is solved by finding the uniqueness of Nash equilibrium.

### III. PRELIMINARIES

In this section, we will briefly introduce the fundamental concepts of constrained optimization problems, differential evolution and successive convex approximation.

#### A. CONSTRAINED OPTIMIZATION PROBLEMS

Without loss of generality, COPs in a minimization sense can be formulated as follows:

\[
\min_{x} f(x) \quad \text{s.t.} \quad g_j(x) \leq 0, \quad \forall j = 1, \ldots, m \tag{1a}
\]

\[
h_j(x) = 0, \quad \forall j = m + 1, \ldots, l \tag{1b}
\]

where \(f(x)\) is the objective function that needs to be minimized over the decision variables \(x = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n\) while satisfying the \(m\) inequality constraints and \(l - m\) equality constraints. Note that the equality constraints (1b) can also be converted into inequality constraints with positive tolerance degree \(\delta\) as

\[
|h_j(x)| - \delta < 0. \tag{2}
\]

For COPs, the degree of constraint violation over the decision variable \(x\) can be denoted as

\[
G(x) = \sum_{j=1}^{l} G_j(x), \tag{3}
\]

where \(G_j(x)\) is the degree of constraint violation on the \(j\)th constraint and can be calculated as

\[
G_j(x) = \begin{cases} 
\max\{0, g_j(x)\}, & j = 1, \ldots, m \\
\max\{0, |h_j(x)| - \delta\}, & j = m + 1, \ldots, l 
\end{cases} \tag{4}
\]

Therefore, a candidate solution is called feasible if \(G_j(x) = 0\), for all \(j = 1, \ldots, l\), i.e., all constraints are satisfied. Optimal solutions must be feasible but feasible solutions are not necessarily optimal.

#### B. DIFFERENTIAL EVOLUTION

DE belongs to the category of evolutionary algorithms. Different from other genetic algorithms, DE generates offspring by incorporating the difference between genes from randomly chosen parents in the pool. Generally, the main procedure of DE can be described as following stages.
1) Initialization
DE initializes by selecting a population size of \(NP\), which is the cardinality of the set of decision vectors \(P = \{x_1, \ldots, x_{NP}\}\). For \(i\)th decision vector \(x_i\), the \(j\)th decision variable can be randomly generated as follows:
\[
x_{i,j} = x_{i,j}^{lb} + rand(0,1) \cdot (x_{i,j}^{ub} - x_{i,j}^{lb}),
\]
where \(x_{i,j}^{lb}\) and \(x_{i,j}^{ub}\) represents the lower bound and upper bound for decision variable \(x_{i,j}\), while \(rand(0,1)\) is the uniformly distributed random numbers between 0 and 1.

2) Mutation
After the initialization of population and decision vectors, the individuals in the population will undergo a mutation stage. In this stage, a mutation operation is applied to generate a mutant decision vector for each original decision vector \(x^t_i\) (\(i \in S = \{1, \ldots, NP\}\)) at generation \(t\). For simplicity, DE/rand/1 mutation scheme is selected for illustration as follows:
\[
v^t_i = x^t_{r_1} + F \cdot (x^t_{r_2} - x^t_{r_3}),
\]
where \(v^t_i\) is the \(i\)th mutant decision vector at generation \(t\), \(F \in (0,1+\varepsilon)\) is the differential scaling factor and \(r_1, r_2\) and \(r_3\) are randomly generated integers from \(S\) such that \(r_1 \neq r_2 \neq r_3 \neq i\). Other commonly used mutation schemes include DE/rand/2, DE/rand-to-best/1, DE/current-to-best/1 and DE/current-to-rand/1 [32].

3) Crossover
In order to maintain diversity in the population, for the \(i\)th individual at stage \(t\), a trial decision vector \(u^t_i\) is generated between original decision vector \(x^t_i\) and mutant decision vector \(v^t_i\). The crossover operation can be described as follows:
\[
u^t_{i,j} = \begin{cases} 
  v^t_{i,j}, & \text{if } rand(0,1) < CR \text{ or } j = j_0 \\
  x^t_{i,j}, & \text{otherwise}
\end{cases}
\]
where \(u^t_{i,j}\), \(v^t_{i,j}\) and \(x^t_{i,j}\) represents the \(j\)th variable of \(u^t_i\), \(v^t_i\) and \(x^t_i\) at generation \(t\), respectively; \(CR \in [0,1]\) is the crossover rate and \(j_0\) is a randomly generated integer from \(S\).

4) Selection
In this stage, the decision vector of the next generation is determined from the current generation by comparing the fitness (i.e., objective function value) of the trial decision vector and original decision vector. It is important to handle constraints when we apply DE to deal with COPs. Classical feasibility rules [33] to handle constraints are summarized as follows:
- Between two feasible decision vectors, the one with smaller objective function value (minimization sense) is selected.
- If one decision vector is feasible and the other one is infeasible, the feasible one is selected.
- Between two infeasible decision vectors, the one with less degree of constraint violation is selected.

Therefore, the selection operation for COPs with positive tolerance \(\varepsilon\) [34] can be described as follows:
\[
x^t_{i} \leftarrow \begin{cases} 
  u^t_i, & \text{if } G(x^t_i) \leq \varepsilon \text{ and } G(u^t_i) \leq \varepsilon \\
  x^t_i, & \text{if } G(x^t_i) \leq \varepsilon \text{ and } G(u^t_i) > \varepsilon \\
  u^t_i, & \text{if } G(x^t_i) > \varepsilon \text{ and } G(u^t_i) \leq \varepsilon \\
  x^t_i, & \text{if } G(x^t_i) > \varepsilon \text{ and } G(u^t_i) > \varepsilon
\end{cases}
\]

C. SUCCESSIVE CONVEX APPROXIMATION
SCA technique has been broadly applied in many areas such as communications, machine learning, networking and signal processing. In what follows, we will review the specific optimization problem solved by the SCA technique under required assumptions and give some examples of function approximation.

1) Problem Statement
Consider the optimization problem as follows:
\[
\begin{align*}
\min_{x} & \quad f_0(x) \triangleq h_0(x) + g_0(x) \\
\text{s.t.} & \quad f_j(x) \triangleq h_j(x) + g_j(x) \leq 0, \quad \forall j = 1, \ldots, l \\
& \quad x \in \mathcal{X},
\end{align*}
\]
where the function \(f_j(x)\) is smooth (possibly nonconvex) and \(g_j(x)\) is convex (possibly nonsmooth), for all \(j = 0, \ldots, l\) and the feasible set is denoted as \(\mathcal{X}\). A commonly used method for solving this specific problem is SCA (also known as majorization minimization) where under a tight convex restriction of the constraint sets, a locally tight approximation of the original optimization problem is solved iteratively. A tight approximation of the original optimization problem can be stated as follows: given \(x^k \in \mathcal{X}\) at iterate \(k\)
\[
\begin{align*}
\min_{x} & \quad f_0(x;x^k) \\
\text{s.t.} & \quad f_j(x;x^k) \leq 0, \quad \forall j = 1, \ldots, l \\
& \quad x \in \mathcal{X}^k,
\end{align*}
\]
where \(f_0(x;x^k)\) and \(f_j(x;x^k)\) represent the approximated functions of \(f_0(x)\) and \(f_j(x)\), respectively and \(x \in \mathcal{X}^k\) is the feasible set at iterate \(k\). More precisely, the SCA method is presented in Algorithm 1. The key assumptions [35] to validate this algorithm are summarized as follows:

Algorithm 1 SCA Algorithm for Problem (9)
Find a feasible solution \(x^k \in \mathcal{X}\) in (9), choose a step size \(\gamma \in (0,1]\) and set \(k = 0\).

Repeat
1) Compute \(\hat{x}^k\), the solution of (10);
2) Set \(x^{k+1} = x^k + \gamma(\hat{x}^k - x^k)\);
3) Set \(k \leftarrow k + 1\).

Until some convergence criterion is met.
Assumption 1: The approximated functions $\tilde{f}_j(\bullet; \bullet)$ for all $j = 0, \ldots, l$ are assumed to satisfy the following statements:

- $\tilde{f}_j(x; y)$ is continuous in $(x, y)$ for all $x, y \in \mathcal{X}$.
- $\tilde{f}_j(\bullet; \bullet)$ is convex for all $y \in \mathcal{X}$.
- $f_j(x; y) = \tilde{h}_j(x; y) + g_j(x)$ for all $x, y \in \mathcal{X}$.
- Function value consistency: $\tilde{h}_j(x; x) = h_j(x)$ for all $x \in \mathcal{X}$.
- Gradient consistency: $\nabla_x \tilde{f}_j(\bullet; x) = \nabla_x f_j(x)$ for all $x \in \mathcal{X}$, where $\nabla_x \tilde{f}_j(\bullet; x)$ denotes the partial gradient of the function $\tilde{f}_j(\bullet; x)$ with respect to the argument $x$ evaluated at $(\bullet; x)$.
- Upper bound: $f_j(x; y) \geq \tilde{f}_j(x)$ for all $x, y \in \mathcal{X}$.

Therefore, the above assumptions guarantee that the original functions can be approximately replaced by suitable upper-bounding functions where the same first-order behavior can be preserved.

2) Examples of Function Approximation

In this section, we will briefly introduce some examples of function approximation technique that will be used throughout this paper.

**Example 1—Approximation of $f_0(x)$:** (Example 8 in [4]) Suppose that $f_0(x)$ has a product of functions (PF) structure, i.e., $f_0(x) = p_1(x)p_2(x)$, with both $p_1$ and $p_2$ being positive and convex. For any $y \in \mathcal{X}$, a convex approximation of $f_0(x)$ is given by

$$\tilde{f}_0(x; y) = p_1(x)p_2(y) + p_1(y)p_2(x) + \frac{\tau}{2}(x - y)\top U(y)(x - y),$$

where $\tau > 0$ is a positive constant, and $U(y)$ is a uniformly positive definite matrix.

**Example 2—Approximation of $f_j(x)$:** (Example 3 in [4]) Suppose that $f_j(x)$ has a difference of convex (DC) structure, i.e., $f_j(x) = f_j^+(x) - f_j^-(x)$ with both $f_j^+(x)$ and $f_j^-(x)$ being continuously differentiable and convex. By linearizing the concave part $-f_j^-(x)$, we obtain a convex upper approximation of $f_j(x)$ as follows: for all $x, y \in \mathcal{X}$,

$$\tilde{f}_j(x; y) \triangleq f_j^+(x) - f_j^-(y) - \nabla_x f_j^-(y)\top(x - y) \geq f_j(x).$$

**Example 3—Approximation of $f_j(x)$:** (Example 4 in [4]) Suppose that $f_j(x)$ has a PF structure, i.e., $f_j(x) = q_1(x)q_2(x)$ with both $q_1(x)$ and $q_2(x)$ being positive and convex. Observe that $f_j(x)$ can be rewritten as a function with a DC structure:

$$f_j(x) = \frac{1}{2}(q_1(x) + q_2(x))^2 - \frac{1}{2}(q_1^2(x) + q_2^2(x)).$$

Then, a convex upper approximation of $f_j(x)$ can be obtained by linearizing the concave part in (13): for any $y \in \mathcal{X}$,

$$\tilde{f}_j(x; y) \triangleq \frac{1}{2}q_1(x) + q_2(y)^2 - \frac{1}{2}(q_1^2(y) + q_2^2(y)) - q_1(y)q_1^\prime(y)(x - y) - q_2(y)q_2^\prime(y)(x - y) \geq f_j(x).$$

IV. SCREENED DE-SCA

Motivated by the pros and cons of DE and SCA, we propose to merge their own advantages in jointly optimizing COPs such that better local optimal solutions of the original problem can be obtained. The framework and algorithm of Screened DE-SCA are presented in Figure 1 and Algorithm 2, respectively. In Figure 1, the process of finding and screening DE solutions (step 1–7) can be implemented in parallel to the process of problem reformulation (step 1 and step 8–9). In step 10–12, the screened DE solutions from step 7 will be used to initialize the SCA-based algorithm on reformulated convex problem from step 9, and then the optimal solutions can be found.

A. DE STAGES

Our algorithm starts from applying the DE algorithm. Specifically, DE algorithm spans from line 1 to line 6 where stages of initialization, mutation, crossover and selection that are implemented based on equations from (5) to (8) are applied to solve the original nonconvex problem (9). Note that we selected DE/rand/1 as a mutation scheme and $\varepsilon$ approximation as a constraint-handling technique [36], [37]. One can also choose different mutation schemes and constraint-handling techniques in this stage. However, it is out of the scope of our paper to determine which mutation schemes and constraint-handling techniques work best in our proposed algorithm.

B. SCREENED DE SOLUTIONS

The DE algorithm will simply terminate if the number of iterations exceeds the pre-defined maximum DE iterations $MaxIter$. Then, we can output and store the best solutions $x^{DE}$ as in line 6. Next, we will need to initialize the SCA-based algorithm and the straightforward idea is to initialize...
Algorithm 2 Screened DE-SCA Algorithm for Problem (9)
Input:
DE: NP, F ∈ (0, 1+], CR ∈ [0, 1], δ, ε,
MaxIter: the maximum DE iterations.
SCA: α, γ(k) ∈ (0, 1], τ > 0.
1: Set \( t = 1 \);
2: Randomly generate an initial population \( P^t = \{x_1^t, \ldots, x_N^t\} \) using equation (5);
Repeat:
3: Evaluate the objective functions \( f_0(x_i^t) \) and degree of constraint violation \( G(x_i^t) \) for all \( i \in S \);
4: for \( i = 1: NP \) do
   1) Calculate mutant decision vector \( v_i^t \) using equation (6);
   2) for \( j = 1: ND \) do
   /\( ND \) is the number of decision variables/\n      i) Obtain trial decision variable \( u_{i,j}^t \) using equation (7);
      3) Apply the feasibility rule with tolerance \( \varepsilon \) according to equation (8);
   5: Set \( t \leftarrow t + 1 \);
Until \( t > MaxIter \)
6: Output and store the best individual in \( P^t \) as \( x_{DE}^* \);
7: Set \( k = 0 \);
8: Randomly initialize the SCA-based algorithm with feasible solutions \( x_k \in \mathcal{X} \) where the feasible parts of \( x_{DE}^* \) are directly used for assignment;
Repeat:
9: Compute \( \hat{x}^k \), the solution of (10);
10: Set \( x^{k+1} = x^k + \gamma(\hat{x}^k - x^k) \), with \( \gamma(k) = \gamma(k-1)(1 - \alpha \gamma(k)) \);
11: Set \( k \leftarrow k + 1 \);
Until \( x^k \) is a stationary solution of (10);
12: Output the optimal solution \( x_{SCA}^* \) and objective function value \( f_0(x_{SCA}^*) \).

it with \( x_{DE}^* \). However, direct initialization with full DE solutions can actually result in worse initializations in the SCA-based algorithm. We observe that \( x_{DE}^* \) is infeasible in general as the degree of constraint violation \( G(x_{DE}^*) \) cannot be small enough (e.g., \( \delta = 10^{-5} \)) especially when there are hundreds of constraints. Therefore, it is necessary to go through the screening process of the best DE solutions before we perform the initialization. In real problems, there can be constraints that contain all decision variables as well as constraints that contain only one decision variable. The set of indices of feasible decision variables from the best DE solution can be found as

\[ K = \{ z \in Z | G_j(x_{DE}^*) = 0, \forall j = 1, \ldots, l \}, \]

where \( Z = \{1, \ldots, ND\} \) denotes the set of indices for decision variables; \( G_j(x_{DE}^*) \) denotes the \( z \)th decision variable from the best DE solution. In order for the \( z \)th decision variable to be considered feasible, we require that all the constraints that contain it must have degree of constraint violation with 0. Therefore, the feasible parts of DE solutions can be denoted as \( x_{DE}^{K} \), while the infeasible parts are simply denoted as \( x_{DE}^{\bar{K}} \).

C. SCA STAGES
While we apply DE algorithm to solve for the original nonconvex problem, we also need to convert the original problem into its approximated convex problem such that the SCA-based algorithm can be applied to find the feasible and optimal solutions of it. Specifically, SCA-based algorithm spans from line 7 to line 12 where the approximated convex problem (10) is iteratively solved with a random solution \( x^0 \) as initialization. It is meaningful to only use the feasible parts \( x_{DE}^{K} \) of the best DE solution to initialize the SCA-based algorithm in line 8 as the initialization must be feasible over all decision variables \( x \in \mathcal{X} \). Note that at line 10, in order to improve the convergence rate of the SCA-based algorithm, a diminishing step-size rule is applied compared to a constant step size in Algorithm 1. To terminate the SCA-based algorithm by finding the stationary solution of (10), we can simply check if \( \|\hat{x}^k - x^k\| \leq \zeta \) where \( \zeta \) is the desired algorithm accuracy. Finally in line 12, we output the optimal solution \( x_{SCA}^* \) and plug it into the original objective function to find \( f_0(x_{SCA}^*) \).

V. UAV-ENABLED MOBILE EDGE COMPUTING
In this section, we will briefly introduce the proposed model in [38] used to validate our innovative Screened DE-SCA algorithm. We first present the system model for a UAV-enabled IoT task offloading process. Then, a nonconvex optimization problem is formulated to model this process.

A. SYSTEM MODEL
In this paper, we consider a UAV-enabled IoT task offloading process as illustrated in Figure 2, which reflects a three-tier network infrastructure: 1) User layer consists of a set of ground IoT devices \( i \in \mathcal{N} = \{1, \ldots, N\} \) which have periodical computation-intensive tasks to perform; 2) UAV layer consists of a UAV that is equipped with CCS resources but is constrained by SWAP; 3) Edge layer consists of a set of ground ECs \( j \in \mathcal{J} = \{1, \ldots, J\} \). ECs are composed of edge servers co-located with base stations or access points, which are deployed in the proximity of IoT devices.

We assume that the IoT devices do not perform local computing due to their limited computational capacities and thus the generated tasks need to be offloaded to the ECs that have more resources for processing. We consider the scenario that these IoT devices cannot directly communicate with ground ECs due to terrestrial signal blockage and shadowing but a UAV can be deployed to help facilitate the task offloading process via unhindered ground-to-air (G2A) and air-to-ground (A2G) communications due to its high altitude.

1) Task Model
The generated task for each IoT device \( i \) can be modeled as a triplet \( \mathcal{M}_i = \{L_i, C_i, \lambda_i\} \) where \( L_i \) denotes the input data size in bits, \( C_i \) denotes the required CPU cycles to process 1
bit of data and \( \lambda_i \) denotes the number of tasks generated per second.

2) Coordinate Model

Our model can be visualized in a 3D Cartesian coordinate system where the locations of ground IoT devices, UAV and ground ECs can be denoted as \( O_i^M = (x_i^M, y_i^M, 0) \), \( O^U = (x^U, y^U, H) \) and \( O_j^E = (x_j^E, y_j^E, 0) \), respectively. We assume the height \( H \) of the UAV is fixed but the horizontal locations \( x^U \) and \( y^U \) of the UAV need to be optimized in our problem since the deployment of the UAV will affect the channel gain during G2A and A2G communications.

3) Communication Model

The G2A uplink channel gain from IoT device \( i \) to the UAV can be obtained using the free-space path loss model
\[
\beta_i^{ul} = \frac{\beta_0}{d_{i,ul}^2} = \frac{\beta_0}{\|O_i^M - O^U\|^2}, \tag{16}
\]
where \( \beta_0 \) represents the received power at the reference distance of 1 m for a transmission power of 1 W, \( d_{i,ul} \) denotes the distance from IoT device \( i \) to the UAV, and \( \| \cdot \| \) denotes the Euclidean norm of a vector. Accordingly, the A2G downlink channel gain from the UAV to EC \( j \) can be obtained as
\[
\beta_j^{dl} = \frac{\beta_0}{d_{j,dl}^2} = \frac{\beta_0}{\|O^U - O_j^E\|^2}, \tag{17}
\]
where \( d_{j,dl} \) denotes the distance from the UAV to EC \( j \).

Assume a FDMA protocol is applied for bandwidth sharing in our studied model. Then, the achievable uplink transmission data rate in bps from IoT device \( i \) to the UAV can be calculated according to Shannon–Hartley theorem as
\[
D_i^{ul} = B_i^{ul} \log_2 \left( 1 + \frac{g_i^{ul} P_i^M}{\sigma^2} \right), \tag{18}
\]
where \( B_i^{ul} \) denotes the allocated bandwidth to IoT device \( i \), \( P_i^M \) denotes the transmission power of IoT device \( i \) and \( \sigma^2 \) denotes the noise power at the UAV. Without loss of generality, we assume both UAV and ECs have the same noise power [39]. Accordingly, the achievable downlink transmission data rate in bps from the UAV to EC \( j \) can be calculated as
\[
D_j^{dl} = B_j^{dl} \log_2 \left( 1 + \frac{g_j^{dl} P_{TX}}{\sigma^2} \right), \tag{19}
\]
where \( B_j^{dl} \) denotes the per-device bandwidth pre-allocated to the UAV when it communicates with EC \( j \) and \( P_{TX} \) denotes the transmission power of UAV.

4) Delay Model

In this model, the total delay incurred during the IoT task offloading process can be divided into: i) G2A uplink transmission delay; ii) computation delay at the UAV; iii) A2G downlink transmission delay; and iv) computation delay at ECs.

**G2A Uplink Transmission Delay**: As mentioned before, IoT devices are assumed to not perform any local computing due to their limited computational capacities and thus all tasks will be first offloaded to the UAV. The G2A uplink transmission delay from IoT device \( i \) can be calculated as
\[
t_i^{G2A} = L_i / D_i^{ul}. \tag{20}
\]

**Computation Delay at the UAV**: After receiving all the tasks from IoT device \( i \), the UAV will determine the task partitioning where the portion of \( \alpha_{ij0} \in [0, 1] \) will be processed at the UAV while the portion of \( \alpha_{ij} \in [0, 1] \) will be further offloaded to EC \( j \) for processing. The computation delay at the UAV when processing the offloaded tasks from IoT device \( i \) can be calculated as
\[
t_i^{U} = \frac{\alpha_{ij0} L_i C_i}{f_i^{U}}, \tag{21}
\]
where \( f_i^{U} \) denotes the allocated computation capacities in CPU cycles/s from the UAV to IoT device \( i \).

**A2G Downlink Transmission Delay**: The A2G downlink transmission delay of task offloading from IoT device \( i \) to EC \( j \) via the UAV can be calculated as
\[
t_{ij}^{A2G} = \frac{\alpha_{ij} L_i}{D_j^{dl}}. \tag{22}
\]

**Computation Delay at ECs**: The computation delay at ECs when processing the tasks offloaded from IoT device \( i \) to EC \( j \) via the UAV can be calculated as
\[
t_{ij}^{E} = \frac{\alpha_{ij} L_i C_i}{f_i^{E}}, \tag{23}
\]
where $f_{ij}^U$ denotes the allocated computation capacities in CPU cycles/s from EC $j$ to IoT device $i$.

Finally, the total delay experienced by IoT device $i$ during task offloading can be calculated as

$$T_i = t_{iG2A}^U + \max_{j \in J}\{t_{ij}^U, t_{ij}^{A2G} + t_{ij}^E\}. \quad (24)$$

5) Energy Model

In this model, we mainly consider the energy consumed at the UAV side during computation and communication processes since the battery size of the UAV is limited.

*Computation Energy Consumption:* The power consumption of the CPU in the UAV when processing tasks offloaded from IoT device $i$ can be modeled as $\kappa(f_{ij}^U)^3$ according to [40], where $\kappa$ denotes the effective switched capacitance based on the CPU architecture. Therefore, the energy consumption of the UAV when processing tasks offloaded from IoT device $i$ can be calculated as

$$E_{i}^{CP} = \kappa(f_{ij}^U)^3 t_{ij}^U = \kappa \beta_{ij} L_i C_i (f_{ij}^U)^2. \quad (25)$$

*Communication Energy Consumption:* The reception energy consumption of the UAV when receiving the task input data from IoT device $i$ can be calculated as

$$E_{i}^{RX} = P_{RX}^U t_{iG2A}^U = L_i P_{RX}^U \frac{D_{iul}}{D_{ij}^T}. \quad (26)$$

where $P_{RX}^U$ denotes the receiving power of UAV. Similarly, the transmission energy consumption of the UAV when transmitting the task input data of IoT device $i$ from the UAV to EC $j$ can be calculated as

$$E_{ij}^{TX} = P_{TX}^U t_{ij}^{A2G} = \alpha_{ij} L_i P_{TX}^U \frac{D_{ij}^T}{D_{ij}^T}. \quad (27)$$

Finally, the total energy consumption of the UAV when serving the computation and communication needs of IoT device $i$ during task offloading can be calculated as

$$E_i = \lambda_i (E_{i}^{CP} + E_i^{RX} + \sum_{j \in J} E_{ij}^{TX}). \quad (28)$$

B. PROBLEM FORMULATION

Based on the system model proposed above, our problem can be stated as follows: with the objective of minimizing the weighted sum of UAV energy consumption and system latency experienced by all IoT devices, we jointly optimize the UAV position $O^U$, G2A communication resource allocation $B_{ij}^{ul}$, task partitioning $\alpha_{ij}$ and $\alpha_{i0}$ and computation resource allocation of the UAV $f_{ij}^U$ and ECs $f_{ij}^E$. It can be formulated as the following optimization problem:

$$\min_{O^U, B_{ij}^{ul}, \alpha_{ij}, f_{ij}^U, f_{ij}^E} \sum_{i \in N} E_i + \varrho \sum_{i \in N} T_i \quad (29a)$$

s.t.

$$\sum_{i \in N} B_{ij}^{ul} \leq B^{ul} \quad (29b)$$

$$\alpha_{i0} + \sum_{j \in J} \alpha_{ij} = 1, \forall i \quad (29c)$$

$$\sum_{i \in N} f_{ij}^U \leq F^U \quad (29d)$$

$$\sum_{i \in N} f_{ij}^E \leq F^E, \forall j \quad (29e)$$

$$0 \leq \alpha_{ij} \leq 1, \forall i, j \quad (29f)$$

$$0 \leq \alpha_{i0} \leq 1, \forall i \quad (29g)$$

$$B_{ij}^{ul} f_{ij}^U \geq 0, \forall i \quad (29h)$$

$$f_{ij}^E \geq 0, \forall i, j \quad (29i)$$

where $\varrho$ is a positive constant describing the relative weight of UAV energy consumption and system latency; (29b), (29d), (29e), (29h) and (29i) guarantee that the allocated resources for G2A bandwidth, CPU frequencies of UAV and ECs are non-negative and cannot exceed their limits $B^{ul}$, $F^U$ and $\{F^E\}_{j=1}^J$, respectively; (29c), (29f) and (29g) ensure that the offloading tasks of all IoT devices are completely partitioned over the UAV and ECs, and each partition variable is between 0 and 1.

Problem (29) is highly nonconvex due to the nonconvex objective function (29a). To tackle the nonconvexity, function approximation methods in Section III-C2 are applied to convert them into suitable convex substitutes since all delay terms $t_{iG2A}^U$, $t_{ij}^U$, $t_{ij}^{A2G}$ and $t_{ij}^E$ and energy terms $E_{i}^{CP}$, $E_{i}^{TX}$ and $E_{ij}^{TX}$ have a PF structure. In this paper, we will skip the reformulation process of problem (29) since our focus is mainly on the solution rather than the system modeling. However, interested readers can refer to Section IV in [38] for the detailed derivation.

VI. NUMERICAL EXPERIMENTS

In this section, we will validate the effectiveness of our proposed Screened DE-SCA algorithm via extensive numerical experiments. All the experiments are implemented in MATLAB R2019b on a desktop computer with an Intel Core i7-8700 3.20GHz CPU and 32GB RAM.

A. SIMULATION SETUP

For simplicity, we consider the same simulation settings used in [38] where 4 ground ECs are placed at each vertex and 10 ground IoT devices are randomly distributed in a 2D area of $1 \times 1 \text{km}^2$ as depicted in Figure 3. The simulation parameters are summarized in Table 2 unless otherwise stated.

In order to validate the effectiveness of our proposed Screened DE-SCA algorithm in terms of achieving the best
system cost of problem (29) in the UAV-enable MEC system, we consider the baseline algorithms as follows:

- DE: Extend the classical DE algorithm to solve (29) by utilizing the feasibility rules introduced in Section III-B.
- DE with Penalty Function (DE-PF): Add the degree of constraint violation $G(x)$ to the objective function as a penalty function when applying DE to solve (29).
- DE with Initialization Normalization (DE-IN): Force the equality constraints (29c) to be satisfied by normalizing the value of initialized task partitioning variables when applying DE to solve (29).
- SCA-based Method: Apply SCA technique to transform problem (29) into approximated convex problem and then iteratively solve it with random initialization.
- SCA-based Method with Full DE Solutions as Initialization (DE-SCA): Initialize the approximated convex problem of (29) by full solutions obtained from solving (29) with DE and then iteratively solve it.
- UMEC: We adapt the task selection and scheduling algorithm proposed in [26] to solve for our nonconvex problem of joint task offloading and resource allocation.
- LCPSO: We adapt the learning-based task sequence and resource allocation algorithm in [28] to solve for our nonconvex problem of joint task offloading and resource allocation.

We would like to clarify that our system setting is different from those in prior works. Therefore, the approaches proposed in prior studies are not directly applicable to our settings. We carefully select two state-of-the-art methods (not DE or SCA-based) UMEC and LCPSO applied in studying the UAV-based MEC systems. Although the problems studied in their papers are different from ours, we are still able to evaluate them by adapting them into our model. To differentiate from the DE-SCA algorithm, our proposed Screened DE-SCA method only uses the partial DE solutions that are feasible to perform initialization of SCA-based method.

### B. EXPERIMENTAL RESULTS

In this section, we will first discuss the tradeoff between feasibility and optimality incurred while applying DE to solve for COPs. Then, we compare two different DE variants DE-PF and DE-IN used in handling the constraints with original DE in our problem. Finally, we compare our proposed Screened DE-SCA method with baseline methods.

1) Feasibility and Optimality Tradeoff for DE

With current simulation settings of 10 IoT devices and 4 ECs, there are 112 constraints in total including 10 equality constraints. A large number of constraints combining with equality constraints generally generally makes it impossible for the DE algorithm to find the optimal solutions within given time limits. Therefore, it is necessary to relax the constraint requirements by introducing the $\varepsilon$ approximation [34] where a large $\varepsilon$ represents more tolerance we allow for constraint violation. Table 3 summarizes how optimal system cost of problem (29) changes as the $\varepsilon$ decreases, where $G_{eq}(x)$ denotes the degree of constraint violation by the equality constraints (29d) and $\eta$ defines the ratio of $G_{eq}(x)$ and $G(x)$. We observe that when $\varepsilon \geq 0.5$, the degree of constraint violation $G(x)$ will be strictly less than $\varepsilon$ but when $\varepsilon < 0.5$, $\varepsilon$ approximation will be violated as $G(x)$ are larger than $\varepsilon$. Moreover, the relationship between $\varepsilon$ and

### TABLE 2. Simulation Parameters

| Parameters | Values | Parameters | Values |
|------------|--------|------------|--------|
| DE: $CR$   | 0.2    | $F$        | [0.2, 0.8] |
| $MaxIter$  | 10000  | $NP$       | 700    |
| $\delta$   | $10^{-4}$ |           |        |
| SCA: $\alpha$ | 0.5    | $\gamma$  | 0.5    |
| $\zeta$    | $10^{-2}$ |         |        |
| UAV-based MEC: $C_i$ | [100, 200] | $\rho$        | 5      |
| CPU cycles/bit |        |           |        |
| $\sigma^2$ | $-100$ dBm | $F_i^E$ | [6, 9] GHz |
| $\beta_0$  | $-50$ dB | $F_i^U$  | 3 GHz   |
| $\kappa$   | $10^{-28}$ [41], $H$ = 0.1 km |
| $\lambda_i$ | 30 tasks/min | $L_i$ | [1, 5] Mbits |
| $P_i^M$    | 0.1 W  | $B_{i,n}$ | 10 MHz  |
| $P_i^{RX}$ | 0.1 W  | $B_{i,d}$ | 0.5 MHz |
| $P_{RX}$   | 1 W    |           |        |

### TABLE 3. System cost vs. $\varepsilon$ for DE Algorithm

| $\varepsilon$ | $G(x)$ | $G_{eq}(x)$ | $\eta \equiv G_{eq}(x)/G(x)$ | System Cost |
|---------------|--------|-------------|-------------------------------|------------|
| 10            | 0.98   | 7.95        | 79.64%                        | 8.10       |
| 5             | 4.94   | 4.49        | 90.76%                        | 18.24      |
| 1             | 0.97   | 0.45        | 46.48%                        | 42.04      |
| 0.5           | 0.46   | 0.20        | 43.95%                        | 64.63      |
| 0.1           | 0.19   | 0.10        | 54.76%                        | 86.87      |
| 0.05          | 0.14   | 0.06        | 44.00%                        | 117.19     |
| 0.01          | 0.17   | 0.08        | 48.81%                        | 149.81     |
optimal system cost is plotted in Figure 4. We observe that the system cost generally decreases as the value of $\varepsilon$ increases since there are less restrictions on meeting all the constraints. In other words, a tradeoff exists between feasibility and optimality for DE algorithm. The less feasible the solutions are, the better system cost we can achieve. However, all solutions under the given $\varepsilon$ values are actually infeasible as $G(x)$ are non-zero. Note that in solving real problems, the feasibility of solutions should be firstly satisfied before considering the optimality. Therefore, the DE algorithm with $\varepsilon$ approximation can generate a worse system cost if we want the feasibility of our solutions to be better considered.

![Figure 4: System cost of problem (29) as a function of $\varepsilon$ for DE algorithm.](image)

2) DE Variants for Constraint Handling

In this section, we will compare the original DE algorithm with its two variants considering different constraint handling techniques in terms of reducing the system cost. For DE-PF method, we obtain the adjusted objective function by augmenting the original objective function with $G(x)$ since in a minimization sense, we want both the objective function value and $G(x)$ to be as small as possible. According to the value of $\eta$ in Table 3, we found that around 58.34% on average of the degree of constraint violation is caused by the equality constraints. Therefore, in DE-IN method, we force the equality constraints to be satisfied after random DE initialization by normalization:

$$
\alpha'_{i0} = \frac{\alpha_{i0}}{\alpha_{i0} + \sum_{j=1}^{J} \alpha_{ij}}, \quad \forall i
$$  \hspace{1cm} (30a)

$$
\alpha'_{ij} = \frac{\alpha_{ij}}{\alpha_{i0} + \sum_{j=1}^{J} \alpha_{ij}}, \quad \forall i, j
$$  \hspace{1cm} (30b)

where we can guarantee that the equality constraints hold for the normalized task partitioning variables $\alpha'_{i0}$ and $\alpha'_{ij}$. Besides, we can also require that the initialized decision variables for all individuals must satisfy all inequality constraints before we proceed towards mutation stage. However, one can easily get trapped in this initialize-check feasibility-reinitialize endless loop, given the large number of constraints considered in our problem. For simplicity, we set $\varepsilon$ to 1 and summarize the system cost comparison results in Table 4. We observe that given $\varepsilon = 1$, DE-IN algorithm has the lowest system cost while DE-PF algorithm has the smallest $G(x)$. According to the feasibility rules, if one decision vector is feasible and the other one is infeasible, the feasible one is selected. Therefore, we will select the solutions of DE-PF since it has the smallest degree of constraint violation among all three methods.

| Methods        | $\varepsilon$ | $G(x)$ | System Cost |
|----------------|---------------|--------|-------------|
| DE             | 1             | 0.97   | 42.04       |
| DE-PF          | 1             | 0.84   | 37.83       |
| DE-IN          | 1             | 1.00   | 22.67       |

3) Effectiveness of Screened DE-SCA

In this section, we will demonstrate the effectiveness of our proposed Screened DE-SCA algorithm in terms of reducing the system cost of problem (29) compared to other baseline methods. We will also verify why our method can lead to better local optimal solutions.

**Cost comparison among Screened DE-SCA methods:** As discussed in Section VI-B1, the solutions obtained by DE algorithm are indeed infeasible no matter how we select the value of $\varepsilon$. Therefore, we cannot initialize the SCA-based algorithm with full DE solutions since it requires the initialization to be feasible. In contrast, we will only choose the partial parts $x_{DE}^*_{i,*,*}$ of the solution found by DE to perform the initialization. To validate why DE-PF method is the best one to initialize the SCA-based method, we compare the system cost of Screened DE-SCA, Screened DE-IN-SCA and Screened DE-PF-SCA where the SCA-based method is initialized by the feasible parts of DE, DE-IN and DE-PF, respectively. As shown in Table 5, the Screened DE-PF-SCA method can achieve the lowest system cost and thus it demonstrates the superiority of DE-PF over DE and DE-IN methods as a means to initialize the SCA-based method.

| Methods                | System Cost | Energy Cost | Delay Cost |
|------------------------|-------------|-------------|------------|
| Screened DE-SCA        | 39.09       | 1.89        | 7.44       |
| Screened DE-IN-SCA     | 44.26       | 2.16        | 8.42       |
| Screened DE-PF-SCA     | 32.32       | 1.68        | 6.13       |

**Initialization feasibility over the iterates:** Apart from initializing the SCA-based method with feasible solution at the first iteration, we also require that the solutions to the approximated convex problem to be feasible at each iteration since this algorithm runs in an iterative fashion until a stationary solution is found. In other words, the found solutions at the $(k - 1)$th iteration are used to initialize the approximated convex problem at the $k$th iteration. Table 6 summarizes the
initialization feasibility comparison between SCA-based and Screened DE-SCA methods, where "#Iterations" denotes the average number of iterations needed to obtain a stationary solution while $\bar{G}(x)$ denotes the average degree of constraint violation over all iterations. We observe that the Screened DE-SCA only takes 11 iterations on average, which is twice faster than the SCA-based method in achieving a stationary solution. Besides, we observe that $\bar{G}(x)$ of our method is approximately zero across all iterates, which implies the solutions found at all iterates are feasible. However, $\bar{G}(x)$ of 0.77 in SCA-based method indicates that some of the iterative solutions found are indeed infeasible. Therefore, we conclude that because of the initialization feasibility guaranteed at each iterate, our proposed method can generate better local optimal solutions compared to the SCA-based method.

**TABLE 6.** Initialization feasibility comparison for SCA-based and Screened DE-SCA algorithms initialized by DE-PF method

| Methods          | #Iterations | $\bar{G}(x)$ |
|------------------|-------------|--------------|
| SCA-based        | 23          | 0.77         |
| Screened DE-SCA  | 11          | $10^{-4}$    |

**Cost comparison with benchmark methods:** The system cost comparison with other benchmark methods are given in Table 7, where the DE-PF method with $\varepsilon = 1$ is selected for initialization as its system cost is comparable to that of the SCA-based algorithm. Note that all three methods SCA-based, DE-SCA and Screened DE-SCA will generate feasible solutions as the SCA technique can transform the original nonconvex problem into a suitable convex surrogate and then it can be successfully solved by optimization software such as Gurobi and CVX. From this table, we observe that the DE-SCA method does not generate a better system cost as the DE method, the time complexity can be denoted as $O(MaxIter * N^P * (N + J))$ where $MaxIter$ and $N^P$ denote the pre-defined maximum DE iterations and population size. Suppose $MaxIter$ and $N^P$ remain constant in our experiments, the time complexity can be reduced to $O(N + J)$, which has linear running time. As for the SCA-based method, the reformulated problem can be solved by the interior-point optimizer such as SeDuMi and the time complexity of this optimizer can be denoted as $O(ND^3)$ where $ND$ is the number of decision variables $3N + 2NJ + 2$. Therefore, the SCA-based method has polynomial running time, which is "tractable" and "fast" according to Cobham’s thesis [43]. In summary, the total time complexity of our proposed method will be $O((N + J) + O(ND^3))$, which is polynomial and thus tractable.

**Hyperparameters:** Both DE and SCA-based methods are dependent on the hyperparameter initialization such as population size, differential scaling factor and crossover rate in DE and step size in SCA-based method. These hyperparameters need to be tuned by lots of trial and error as there is no theoretic evidence on how to calculate the values of them.

**Convex approximation:** We shall note that not all nonconvex optimization problems can be approximated by suitable convex forms. In our problem setting, there are mainly two types of nonconvex constraints that can be represented by either difference of functions or product of functions. According to the function approximation examples in Section III-C2, they can be successfully converted into convex forms. More candidate nonconvex constraints and objective functions can be found in [4], [5].

**Future work:** There are still some open questions on how to determine qualified DE solutions to initialize the SCA-based method. First, we can investigate which mutation scheme and constraint-handling technique will jointly work the best in finding the solutions. Second, we can define different metrics to evaluate the quality of the DE solutions even if the DE algorithm cannot converge within time limits. Third, we can design different schemes to screen the best DE solutions such that the initialization of the SCA-based method will lead to better solutions.

**TABLE 7.** System cost comparison for Screened DE-SCA using feasible parts of DE-PF solution for initialization with other benchmark methods

| Methods          | System Cost | Energy Cost | Delay Cost |
|------------------|-------------|-------------|------------|
| DE-SCA           | 110.76      | 0.87        | 21.98      |
| LCPSO [28]       | 44.49       | 3.44        | 8.21       |
| UMEC [26]        | 38.37       | 2.33        | 7.28       |
| SCA-based        | 34.46       | 1.59        | 6.57       |
| Screened DE-SCA | 32.32       | 1.68        | 6.13       |

4) Discussions

In this section, we will discuss the potential difficulties of applying the screened DE-SCA method and the future work.

**Time complexity:** The total number of decision variables considered in our problem can be calculated as $3N + 2NJ + 2$ where $N$ and $J$ are the number of IoT devices and ECs, respectively. In simulation, we set $N$ to 10 and $J$ to 4 so there are 112 decision variables in total. For the DE method, the time complexity can be denoted as $O(MaxIter * N^P * (N + J))$ where $MaxIter$ and $N^P$ denote the pre-defined maximum DE iterations and population size. Suppose $MaxIter$ and $N^P$ remain constant in our experiments, the time complexity can be reduced to $O((N + J))$, which has linear running time. As for the SCA-based method, the reformulated problem can be solved by the interior-point optimizer such as SeDuMi and the time complexity of this optimizer can be denoted as $O(ND^3)$ where $ND$ is the number of decision variables $3N + 2NJ + 2$. Therefore, the SCA-based method has polynomial running time, which is "tractable" and "fast" according to Cobham’s thesis [43]. In summary, the total time complexity of our proposed method will be $O((N + J)) + O(ND^3)$, which is polynomial and thus tractable.

**VII. CONCLUSION**

In this article, we have proposed an innovative Screened DE-SCA algorithm by jointly considering both the classical DE algorithm and SCA-based algorithm. This algorithm can not only overcome the lack of feasibility guarantee in DE algorithm but also the inability of working on original nonconvex problems in SCA-based algorithm. Specifically, this algorithm unifies DE and SCA techniques such that the feasible parts of the solutions of an original nonconvex problem found by DE algorithm can be used to initialize the SCA-based algorithm on its approximated convex surrogate. We have utilized a UAV-enabled MEC system that involves the interactions among IoT devices, UAV and ECs to validate...
the effectiveness of our proposed algorithm as the formulated problem therein is highly nonconvex. Through extensive experiments, we have verified that our proposed Screened DE-SCA algorithm largely outperforms benchmarks including DE, SCAs-based and state-of-the-art algorithms to solve the formulated problem in the UAV-enabled MEC system. In the future, we will extend our proposed algorithm to solve the optimization problems incurred in a UAV-enabled MEC system that contains multiple UAVs.

REFERENCES

[1] S. A. Huda and S. Moh, “Survey on computation offloading in UAV-enabled mobile edge computing,” Journal of Network and Computer Applications, p. 103341, 2022.
[2] Y. Yazid, I. Er-Zazzi, A. Guerrero-González, A. El Oualkadi, and M. Arjoua, “UAV-enabled mobile edge-computing for IoT based on AI: A comprehensive review,” Drones, vol. 5, no. 4, p. 148, 2021.
[3] B. R. Marks and G. P. Wright, “A general inner approximation algorithm for nonconvex mathematical programs,” Operations research, vol. 26, no. 4, pp. 681–683, 1978.
[4] G. Scutari, F. Facchinei, and L. Lampariello, “Parallel and distributed methods for constrained nonconvex optimization—part I: Theory,” IEEE Transactions on Signal Processing, vol. 65, no. 8, pp. 1929–1944, 2017.
[5] G. Scutari, F. Facchinei, L. Lampariello, S. Sardellitti, and P. Song, “Parallel and distributed methods for constrained nonconvex optimization—part II: Applications in communications and machine learning,” IEEE Transactions on Signal Processing, vol. 65, no. 8, pp. 1945–1960, 2016.
[6] R. Storn and K. Price, “Differential evolution—a simple and effective heuristic for global optimization over continuous spaces,” Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997.
[7] T. Back, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, 1996.
[8] W. Gong, Z. Cai, and D. Liang, “Adaptive ranking mutation operator based differential evolution for constrained optimization,” IEEE Transactions on Cybernetics, vol. 45, no. 4, pp. 716–727, 2014.
[9] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. C. Coello, and K. Deb, “Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization,” Journal of Applied Mechanics, vol. 41, no. 8, pp. 8–31, 2006.
[10] M. H. Mousa and M. K. Hussein, “Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization,” PeerJ Computer Science, vol. 8, p. e870, 2022.
[11] M. Asim, W. K. Mashwani, H. Shah, and S. B. Belhaurai, “An evolutionary trajectory planning algorithm for multi-UAV-assisted MEC system,” Soft Computing, pp. 1–14, 2021.
[12] W. Feng, J. Tang, N. Zhao, X. Zhang, X. Wang, and K.-K. Wong, “A deep learning-based approach to resource allocation in UAV-aided wireless powered MEC networks,” in ICC 2021 IEEE International Conference on Communications. IEEE, 2021, pp. 1–6.
[13] P.-q. Huang, Y. Wang, and K.-z. Wang, “Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system,” Frontiers of Information Technology & Electronic Engineering, vol. 21, no. 12, pp. 1713–1725, 2020.
[14] B. Xu, L. Zhang, Z. Xu, Y. Liu, J. Chai, S. Qin, and Y. Sun, “Energy optimization in multi-UAV-assisted edge data collection system,” Computers, Materials & Continua, 2021.
[15] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, “Multi-UAV-enabled load-balance mobile-edge computing for IoT networks,” IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6898–6908, 2020.
[16] P.-Q. Huang, Y. Wang, K. Wang, and K. Yang, “Differential evolution with a variable population size for deployment optimization in a UAV-assisted IoT data collection system,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 4, no. 3, pp. 324–335, 2020.
[17] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, “Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing,” IEEE Transactions on Cybernetics, vol. 50, no. 9, pp. 3984–3997, 2019.
[18] Y. Qu, H. Dai, H. Wang, C. Dong, F. Wu, S. Guo, and Q. Wu, “Service provisioning for UAV-enabled mobile edge computing,” IEEE Journal on Selected Areas in Communications, vol. 39, no. 11, pp. 3287–3303, 2021.
[19] Y. Xu, T. Zhang, Y. Liu, D. Yang, L. Xiao, and M. Tao, “UAV-assisted MEC networks with aerial and ground cooperation,” IEEE Transactions on Wireless Communications, vol. 20, no. 12, pp. 7712–7727, 2021.
[20] W. Lu, Y. Ding, Y. Gao, S. Hu, Y. Wu, N. Zhao, and Y. Gong, “Resource and trajectory optimization for secure communications in dual unmanned aerial vehicle mobile edge computing systems,” IEEE Transactions on Industrial Informatics, vol. 18, no. 4, pp. 2704–2713, 2021.
[21] Y. Shi, Y. Xia, and Y. Gao, “Joint gateway selection and resource allocation for cross-tier communication in space-air-ground integrated IoT networks,” IEEE Access, vol. 9, pp. 4303–4314, 2020.
[22] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, “Joint computation and communication design for UAV-assisted mobile edge computing in IoT,” IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5505–5516, 2019.
[23] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, “Completion time and energy optimization in the UAV-enabled mobile-edge computing system,” IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7808–7822, 2020.
[24] Z. Wang, W. Xu, D. Yang, and J. Lin, “Joint trajectory optimization and user scheduling for rotary-wing UAV-enabled wireless powered communication networks,” IEEE Access, vol. 7, pp. 181369–181380, 2019.
[25] S. Jeong, O. Simeone, and J. Kang, “Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning,” IEEE Transactions on Vehicular Technology, vol. 67, no. 3, pp. 2049–2063, 2018.
[26] Z. Qin, H. Wang, Z. Wei, Y. Qu, F. Xiong, H. Dai, and T. Wu, “Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities,” IEEE Internet of Things Journal, vol. 8, no. 24, pp. 17290–17307, 2021.
[27] L. Xie, J. Xu, and R. Zhang, “Throughput maximization for UAV-enabled wireless powered communication networks,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1690–1703, 2018.
[28] L. Sun, L. Wan, and X. Wang, “Learning-based resource allocation strategy for industrial IoT in UAV-enabled MEC systems,” IEEE Transactions on Industrial Informatics, vol. 17, no. 7, pp. 5031–5040, 2020.
[29] K. Zhang, X. Gui, D. Ren, and D. Li, “Energy–latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system,” IEEE Internet of Things Journal, vol. 8, no. 8, pp. 6709–6719, 2020.
[30] A. Asheralieva and D. Niyato, “Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8753–8769, 2019.
[31] S. Yan, M. Peng, and X. Cao, “A game theory approach for joint access selection and resource allocation in UAV assisted IoT communication networks,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1663–1674, 2018.
[32] B.-C. Wang, H.-X. Li, J.-P. Li, and Y. Wang, “Composite differential evolution for constrained evolutionary optimization,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 7, pp. 1482–1495, 2018.
[33] K. Deb, “An efficient constraint handling method for genetic algorithms,” Computer Methods in Applied Mechanics and Engineering, vol. 186, no. 2–4, pp. 311–338, 2000.
[34] T. Takahama and S. Sakai, “Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites,” in 2006 IEEE International Conference on Evolutionary Computation. IEEE, 2006, pp. 1–8.
[35] M. Razaviyayn, “Successive convex approximation: Analysis and applications,” Ph.D. dissertation, University of Minnesota, 2014.
[36] C. A. C. Coello, “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art,” Computer Methods in Applied Mechanics and Engineering, vol. 191, no. 11–12, pp. 1245–1287, 2002.
[37] J. Lampinen, “A constraint handling approach for the differential evolution algorithm,” in Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600), vol. 2. IEEE, 2002, pp. 1468–1473.
[38] Z. Yu, Y. Gong, S. Gong, and Y. Guo, “Joint task offloading and resource allocation in UAV-enabled mobile edge computing,” IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3147–3159, 2020.
[39] Y. Zeng, R. Zhang, and T. J. Lim, “Throughput maximization for UAV-enabled mobile relaying systems,” IEEE Transactions on Communications, vol. 64, no. 12, pp. 4983–4996, 2016.
[40] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268–4282, 2016.

[41] W. Yuan and K. Nahrstedt, “Energy-efficient soft real-time CPU scheduling for mobile multimedia systems,” in ACM SIGOPS Operating Systems Review, vol. 37, no. 5, 2003, pp. 149–163.

[42] ——, “Energy-efficient CPU scheduling for multimedia applications,” ACM Transactions on Computer Systems (TOCS), vol. 24, no. 3, pp. 292–331, 2006.

[43] O. Goldreich, “Computational complexity: a conceptual perspective,” ACM Sigact News, vol. 39, no. 3, pp. 35–39, 2008.

ZHE YU (Graduate Student Member, IEEE) received the B.Eng. degree in Communication Engineering from University of Science and Technology Beijing, Beijing, China, in 2014, and the M.S. degree in Electrical Engineering from Vanderbilt University, Nashville, TN, USA, in 2016, respectively. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering at Oklahoma State University, Stillwater, OK, USA, where he works on resource management for energy-efficient cloud and edge systems. He was the recipient of travel grant of the 8th ACM International Conference on Future Energy Systems (e-Energy).

GUOLIANG FAN (Senior Member, IEEE) received the B.S. degree in Automation Engineering from the Xi’an University of Technology, Xi’an, China, the M.S. degree in Computer Engineering from Xidian University, Xi’an, China, and the Ph.D. degree in Electrical Engineering from the University of Delaware, Newark, DE, USA, in 1993, 1996, and 2001, respectively.

He is the Cal and Marilyn Vogt Professor in Engineering with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. His research interests include Image Processing, Pattern Recognition, Machine Learning and Computer Vision.

Dr. Fan is a senior member of IEEE and he served as an Associated Editor for IEEE Transactions on Image Processing from 2014-2018. Currently, he is an Associate Editor for the IEEE Journal of Biomedical and Health Informatics and EURASIP Journal on Image and Video Processing. He was the recipient of the Young Alumni Achievement Award from the Department of Electrical and Computer Engineering, University of Delaware in 2015.