Reinforcement Learning-based Collision Avoidance and Optimal Trajectory Planning in UAV Communication Networks

Yu-Hsin Hsu and Rung-Hung Gau
Institute of Communications Engineering
National Chiao Tung University
Hsinchu, Taiwan
e-mail: mm84080881@gmail.com; runghunggau@g2.nctu.edu.tw

Abstract—In this paper, we propose a reinforcement learning approach of collision avoidance and investigate optimal trajectory planning for unmanned aerial vehicle (UAV) communication networks. Specifically, each UAV takes charge of delivering objects in the forward path and collecting data from heterogeneous ground IoT devices in the backward path. We adopt reinforcement learning for assisting UAVs to learn collision avoidance without knowing the trajectories of other UAVs in advance. In addition, for each UAV, we use optimization theory to find out a shortest backward path that assures data collection from all associated IoT devices. To obtain an optimal visiting order for IoT devices, we formulate and solve a no-return traveling salesman problem. Given a visiting order, we formulate and solve a sequence of convex optimization problems to obtain line segments of an optimal backward path for heterogeneous ground IoT devices. We use analytical results and simulation results to justify the usage of the proposed approach. Simulation results show that the proposed approach is superior to a number of alternative approaches.

Index Terms—reinforcement learning, UAV collision avoidance, optimal trajectory planning, convex optimization, traveling salesman problem with neighborhood.

1 Introduction

Unmanned aerial vehicles (UAVs) have been used for various applications such as surveying, smart agriculture, and cargo delivery. In addition, due to their maneuverability and the existence of line-of-sight (LoS) air-to-ground communication links, UAVs could serve as mobile aerial base stations for expanding network coverage and enhancing system throughput [1].

Typically, after delivering a package to its destination, a UAV has to fly back to the cargo distribution center. In this paper, we propose using UAVs to deliver goods and collect data from ground Internet of Things (IoT) devices. Specifically, a UAV delivers a package of goods from the cargo distribution center to its destination via the forward path, while it collects data from ground IoT devices through the backward path.

When there are multiple UAVs in the network, it is paramount to avoid collisions among UAVs for flight safety. Without centralized control, the UAV collision avoidance problem is essentially an optimal sequential decision problem, which contains lots of unknown dynamics. Therefore, we take an approach of reinforcement learning [2] to attack the UAV collision avoidance problem.

In addition, for each UAV, we aim to minimize the length of the corresponding backward path/trajectory for collecting data from associated ground IoT devices. To collect data from an IoT device, a UAV has to fly close enough to the IoT device, since the transmit power is bounded. As a result, the optimal trajectory planning problem is similar to the traveling salesman problem with neighborhood (TSPN), which is NP-hard. He *et al.* [3] proposed the combine-skip-substitute (CSS) algorithm for using a mobile element (ME) to collect data in wireless sensor networks. The CSS algorithm solves a traveling salesman problem to determine the visiting schedule of the ME and uses circumcircles to combine visiting points. Kim *et al.* [4] introduced the k-traveling salesman problem with neighborhood (k-TSPN), whose goal is to find k tours for data MULEs to cover sensors in the network. Kim *et al.* [5] investigated optimal trajectory planning of multiple drones in search-and-reconnaissance operations. They identified new variants of the TSPN and proposed novel approximation algorithms for solving them.

There are three major differences between the studied optimal trajectory planning problem and similar problems in the literature. First, unlike the traditional TSPN, the studied optimal trajectory planning problem has different start point and termination point, which are both specified in advance. Thus, one cannot solve the optimal trajectory planning problem by solving a TSPN. Second, while previous works [3] [4] [5] assume that all site neighborhoods have the same size, we study the general case in which ground IoT devices might have neighborhoods of different sizes according to their data rate requirements. Since the radii of site neighborhoods in the studied optimal trajectory planning problem could be distinct real numbers, optimization tools in the previous works such as circumcircles [3] cannot be directly applied to the studied problem. Third, a primary design goal of [4] [5] is to obtain k tours so that each sensor in the network is covered by at least one tour, while our design goal is to obtain k tours so that the kth tour covers the kth set of sensors specified in advance.

Chen *et al.* [6] studied and solved the sweep coverage problem under the assumption that the communication areas of targets are disjoint. In contrast, we do not make the assumption. While they focused on the case in which the start point and the termination point of a UAV trajectory are the same, we study the case in which the start point and the termination point of the backward path are different. To obtain an optimal intermediate point for collecting data and minimizing the path length, they use a geometric approach, while we use a convex optimization approach to benefit from modern theory and software. Zeng *et al.* [7] aimed to design an optimal UAV trajectory that minimizes the completion time of delivering a file to a multicast group of ground terminals. They proposed using an optimal set of waypoints to create line segments of a UAV trajectory. In addition, given a set of waypoints, they adopted linear programming to obtain the optimal UAV speed. We focus on using UAVs for collecting distinct data from ground IoT devices in this paper.

Our major technical contributions include the following. First, we propose a distributed reinforcement learning approach for collision avoidance in UAV communication networks. In particular, to significantly reduce the size of the state space and avoid the curse of dimensionality, during the learning process, a UAV only learns and responds to local network states in its neighborhood. In addition, we study the optimal path design problem for a heterogeneous UAV network where ground IoT devices have different communication radii. To determine an optimal backward path for a UAV, we take a two-phase approach that contains combinatorial optimization problems and convex optimization problems. To obtain an optimal order of visiting ground IoT devices, we formulate and solve a no-return traveling salesman problem. For each UAV, given an optimal visiting sequence of IoT devices, we formulate and solve convex optimization problems to obtain the line segments of an optimal backward path. We use analytical results and large-scale simulation results to support the usage of the proposed approach. After learning, UAVs are able to avoid flight collisions. In terms of the total trajectory length, the proposed approach significantly outperforms a number of alternative approaches. To the best of our knowledge, our work is the first in the literature that rigorously studies optimization problems for using UAVs to deliver goods in the forward paths and collect data from heterogeneous ground IoT devices in the backward paths.

The rest of the paper is organized as follows. In Section 2, we briefly introduce related works in the literature. In Section 3, we include system model and problem formulation. In Section 4, we propose a novel trajectory planning algorithm for UAVs. The proposed algorithm is based on combinatorial optimization and convex optimization. In Section 5, we propose a reinforcement learning approach for UAV collision avoidance. We show simulation results in Section 6. Our conclusions are included in Section 7.

2 RELATEDWORK

Collision avoidance is an important issue for UAVs. Lin *et al.* [8] designed and implemented sampling-based path planning methods for a UAV to avoid collision with commercial aircraft and other moving obstacles based on random tree algorithms. Mahjri *et al.* [9] proposed an analytical framework for a three-dimensional conflict detection. They proposed the SLIDE algorithm for conflict detection so that UAVs could safely share a common airspace. Machine learning has been used to solve a number of problems in wireless communication networks including collision avoidance for UAVs. Wang *et al.* [10] proposed a deep reinforcement learning-based method that allows UAVs to execute navigation tasks such as goods delivery and remote surveillance. They formulated the studied problem as a partially observable Markov decision process (POMDP) and solved it by a novel online deep reinforcement learning (DRL) algorithm. The UAV navigation problem studied in [10] is different from the UAV collision avoidance problem studied in the paper. While they focus on static obstacles that do not move, we concentrate on the scenario in which adjacent UAVs with unknown flying trajectories are the only obstacles for a UAV. To significantly reduce the size of the state space and the computational complexity, we do not use any POMDP and take a distributed approach of reinforcement learning for UAV collision avoidance. To reduce the computational complexity, unlike [10], our proposed approach does not use any neural network. We try the reduce the size of the state space by design rather than using deep neural networks to approximate the mapping between states and optimal actions. Lin *et al.* [11] proposed using recurrent neural networks for optimal proactive edge caching in wireless small cell networks. In this paper, we propose using reinforcement learning for collision avoidance in UAV communication networks.

UAVs could be used for supporting uplink communication and collecting data from ground devices. Mozaffari *et al.* [12] studied the problem of collecting IoT data by UAVs. They focused on optimizing the UAV locations in order to minimize the total transmit power of IoT devices. Yang *et al.* [13] focused on dispatching a UAV to collect data from fixed ground terminals (GTs). Specifically, they studied the tradeoff between the energy consumption of a UAV due to flying and that of a ground terminal due to data transmission. They found the optimal GT transmit power and UAV trajectory that achieve Pareto optimal tradeoffs. Yuan *et al.* [14] studied the optimal robot routing problem in which a mobile robot is required to visit all given sensors in the plane for downloading the data and finally return to its base. To minimize the traveling distance of the robot, they formulated a TSPN, which is NP-hard. We formulate a no-return TSPN for optimizing the backward path of a UAV in this paper. Li *et al.* [15] proposed using a UAV as a floating relay for users in a wireless cellular network. They obtained an optimal power allocation strategy for each user and designed two effective online 3-D placement algorithms for the UAV to approach the optimal location. Zhou *et al.* [16] considered the UAV-aided mobile crowd sensing (MCS) system and investigated the associated joint route planning and task assignment problem. In particular, they proposed using dynamic programming to solve the route planning problem and took a game-theoretic approach to deal with the task assignment problem.

UAVs could also support downlink wireless communications. Wu *et al.* [17] proposed maximizing the minimum throughput over all ground users in the downlink communication by jointly optimizing multiuser communication scheduling, user association, UAV trajectories, and UAV power control. Specifically, they designed an efficient iterative algorithm that solves the non-convex optimization problem based on the block coordinate descent and successive convex optimization techniques. Cheng *et al.* [18] investigated using UAVs to serve users at cell edges for data offloading. They focused on optimizing the UAV trajectory for maximizing the sum rate of UAV-served edge users subject to the rate requirements. Given a UAV with initial location and final location, Zhang *et al.* [19] aimed to obtain an optimal UAV trajectory that minimizes the mission completion time subject to quality-of-connectivity constraints. Zhao *et al.* [20] studied the problem of sum rate maximization in a UAV-assisted NOMA wireless network in which the UAV and base stations collaborate to serve ground users. They proposed jointly optimizing the UAV trajectory and the NOMA precoding vectors.

The UAV trajectory design problem is closely related to the geometric coverage problem [21] [22]. For mathematical tractability, it is usually assumed that a trajectory is composed of line segments and therefore is fully characterized by turning points. To minimize the trajectory length, it is desired to select a turning point at which a UAV could serve/cover IoT devices as many as possible. The random geometric disk cover (GDC) algorithm [23] produces an approximate solution for the geometric coverage problem with low computational complexity. Gau *et al.* [24] proposed a dual approach for solving the worst-case-coverage deployment problem in ad-hoc wireless sensor networks. Zhang [25] proposed a polynomial-time approximate algorithm for the maximum lifetime k-coverage problem in wireless sensor networks. In [24] [25], the primary goal is to find optimal locations for deploying sensors or to schedule active/sleeping status of sensors. In contrast, we aim to obtain an optimal trajectory for a UAV to serve a given set of ground IoT devices. More details on coverage problems in wireless sensor networks can be found in [24] [25] and reference therein.

In addition to trajectory design, obtaining optimal final locations for UAVs is also an active research topic. Lyu *at al.* [26] proposed a polynomial-time algorithm for sequentially dispatching UAVs to cover ground terminals. Koyuncu *et al.* [27] proposed a quantization approach for deployment and trajectory optimization of UAVs. For 1-D networks, they determined an accurate formula for the total UAV movement that guarantees the best time-averaged performance. Zhang *et al.* [28] studied two fast UAV deployment problems for optimal wireless coverage. In particular, the first UAV deployment algorithm seeks to minimize the maximum deployment delay among all UAVs for fairness consideration, while the other aims to minimize the total deployment delay for efficiency consideration.

Energy-efficient design is important for UAV communication networks. Zeng *et al.* [29] studied energy-efficient UAV communication with a ground terminal via optimizing the trajectory of a UAV. They assumed that the UAV flies horizontally with a fixed altitude and aimed to optimize the flight radius as well as the speed of the UAV. Li *et al.* [30] proposed using UAVs to extend wireless sensor networks (WSNs) to remote human-unfriendly terrains. They proposed an energy-efficient cooperative relaying scheme which extends the network lifetime while guaranteeing the success rate. Xu *et al* [31] studied a UAV-enabled wireless power transfer system, where a UAV delivers wireless energy to a set of energy receivers at known locations on the ground. Yin *et al.* [32] jointly optimized resource allocation and placement of a wireless-powered UAV in wireless cellular networks. Sun *et al.* [33] investigated 3D trajectory design and resource allocation for solar-powered UAV communication systems. In particular, they aimed to maximize the system sum throughput over a given time period. In this paper, we obtain optimal trajectories for reducing energy consumption and completion time of data collection. Wireless energy transfer and energy harvesting are beyond the scope of the paper.

3 System Model and Problem Formulation

Consider a system that consists of a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of M UAVs and a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of N ground IoT devices. Let $\mathcal{Z}$ be the set of all integers and $\mathcal{R}$ be the set of all real numbers. Let $(x_k, y_k, 0)$ be the coordinates of the location of the kth ground IoT device, $\forall k \in \{1, 2, .., M\}$. Define $\mathbf{w}_k = (x_k, y_k)$, $\forall k$. It is assumed that a ground IoT device is associated with a single UAV. Let A_n be the index of the UAV that serves the ground IoT device n, $\forall n$. Let $\mathcal{K}_m$ be the set composed of indexes of the ground IoT devices that are associated with UAV m, $\forall m$. Namely, $\mathcal{K}_m = \{n \in \mathcal{Z} | 1 \leq n \leq N, A_n = m\}$, $\forall m$.

It is assumed that the speed of a UAV is constant and all UAVs have the same speed. Let V be the speed of a UAV. For each UAV, there are a start point, a destination point, a forward path, and a backward path. Specifically, for a UAV, a forward path is a continuous curve from the start point to the destination point. On the other hand, for a UAV, the backward path is a continuous curve from the destination point to the start point. For a UAV, the forward path is typically different from the backward path, since the two paths are used for different purposes. Each UAV is used for goods delivery and data collection. Specifically, UAV m has to deliver goods from the starting point $q_{start,m} \in \mathcal{R}^3$ to the destination $q_{goal,m} \in \mathcal{R}^3$ through the forward path, $\forall m \in \mathcal{M}$. After delivering goods through the forward path, UAV m has to collect data from the associated ground IoT devices and return to $q_{start,m}$ through the backward path, $\forall m$.

Assume that UAV k flies at altitude h_k after the launching phase, $\forall k$. To efficiently utilize the space, UAVs are partitioned into groups such that $h_i = h_j$ if and only if UAV i and UAV j belong to the same group. Let d_{min} be the minimum distance required for two UAVs to avoid collisions. To assure that two UAVs that belong to different groups never collide, $\min_{(i,j):h_i \neq h_j} |h_i - h_j| > d_{min}$. Since UAVs that belong to different groups do not collide, it is sufficient to focus on a single group. Without loss of

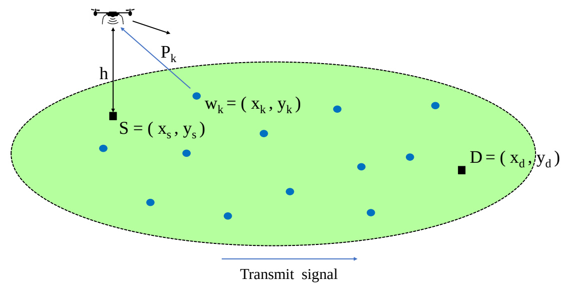


Fig. 1: System model for UAV data collection.

essential generality, it is assumed that all UAVs belong to the same group. As [29], we assume that all UAVs fly at a fixed altitude h after the launching phase.

It is assumed each UAV is equipped with a radar (and/or a lidar). Let d_{sen} be the sensing radius of a UAV. Namely, a UAV detects an adjacent UAV if and only if the distance between the two UAVs is no greater than d_{sen}. To avoid UAV collisions, it is required that $d_{sen} > d_{min}$.

For applications such as using a UAV to deliver a transplanted organ or medicines, it is paramount to deliver goods to the destination as soon as possible. In addition, when goods are heavy, it is detrimental for a battery-powered UAV to carry the goods for too much time. Therefore, for each UAV, we aim to use the forward path to deliver goods as fast as possible without collisions. In addition, we aim to use the backward path to minimize the completion time of collecting data from ground IoT devices without collisions. Later in the paper, we will show how a UAV plans its optimal backward path based on optimization theory and decides its actual paths based on distributed reinforcement learning.

Let $d(S_1, S_2)$ be the distance between two sets S_1 and S_2, where $S_1, S_2 \in \mathcal{R}^3$. In particular,

$$d(S_1, S_2) \quad = \quad \inf_{\mathbf{u} \in S_1, \mathbf{v} \in S_2} \|\mathbf{u} - \mathbf{v}\|. \tag{1}$$

Let $d(\gamma, \mathbf{v})$ be the distance between a differential curve γ and a point $\mathbf{v}$ in the three-dimensional Euclidean space $\mathcal{R}^3$. For a differential curve γ in $\mathcal{R}^3$, let $\{\gamma(\tau) = (\gamma_1(\tau), \gamma_2(\tau), \gamma_3(\tau)) | 0 \le \tau \le 1\}$ be a parameterization. For a differential curve γ in $\mathcal{R}^2$, let $\{\gamma(\tau) = (\gamma_1(\tau), \gamma_2(\tau)) | 0 \le \tau \le 1\}$ be a parameterization.

For each $\mathbf{w} \in \mathcal{R}^2$ and $r > 0$, let $C_{\mathbf{w}}(r)$ be the circle centered at $\mathbf{w}$ with radius r. Namely, $C_{\mathbf{w}}(r) = \{\mathbf{v} \in \mathcal{R}^2 | \, \|\mathbf{v} - \mathbf{w}\| \le r\}$. Let $\partial C_{\mathbf{w}}(r)$ be the boundary of $C_{\mathbf{w}}(r)$. Namely, $\partial C_{\mathbf{w}}(r) = \{\mathbf{v} \in \mathcal{R}^2 | \, \|\mathbf{v} - \mathbf{w}\| = r\}$. Given two sets S_1 and $S_2 \subseteq S_1$, define $S_1 - S_2 = \{x | x \in S_1, x \notin S_2\}$.

3.1 Data collection through backward paths

Since different UAVs serve different ground IoT devices, it is sufficient to focus on a single UAV for the phase of data collection. Thus, $q_{start,m}$ and $q_{goal,m}$ are abbreviated by q_{start} by q_{goal}, respectively. In addition, $q_{start} = (x_s, y_s, h)$ and $q_{goal} = (x_d, y_d, h)$. Furthermore, $\mathcal{K}_m$ is abbreviated by $\mathcal{K}$. Let $K = |\mathcal{K}|$. Without loss of essential generality, it is assumed that $\mathcal{K} = \{1, 2, .., K\}$. Through the backward path, the UAV flies from (x_d, y_d, h) to (x_s, y_s, h) and collects data from IoT devices with indexes in $\mathcal{K}$. In Figure 1, we illustrate the studied communication network that contains UAVs and ground IoT devices. For clarity, we only show a UAV in the figure.

As in many previous works, it is assumed that the air-to-ground (A2G) communication channels are line-of-sight (LoS) links. Namely, the channel gain follows the free space propagation model and depends only on the distance between the UAV and an IoT device. Let d_k be the distance between the UAV and IoT device k, f_c be the carrier frequency, c be the speed of light, and η_{LoS} be the average additional loss for LoS links. According to the LoS channel model [34], $L_{LoS,k}$, the path loss for the link from the UAV to IoT device k in dB is as follows.

$$L_{LoS,k} = 20 \log_{10} \left(\frac{4\pi f_c d_k}{c} \right) + \eta_{LoS}. \tag{2}$$

Let P_k be the transmit power of the kth ground IoT device in dB, $\forall k$. We focus on a heterogeneous network in which the values of P_k's are not the same. Let Γ_k be the threshold of the received SNR in dB for the UAV to successfully decode data from ground IoT device k, $\forall k$. Let P_n be the background noise power in dB. For the UAV to successfully decode data from ground IoT device k, we have the following inequality.

$$P_k - L_{LoS,k} - P_n \ge \Gamma_k. \tag{3}$$

Let $L_k = P_k - P_n - \Gamma_k$ be the path loss threshold in dB, $\forall k \in \mathcal{K}$. Then, $L_{LoS,k} \le L_k$, $\forall k$. Let D_k be the maximum communication distance from ground IoT device k to the UAV. Based on (2), we have

$$L_k \quad = \quad 20 \log \left(\frac{4\pi f_c D_k}{c} \right) + \eta_{LoS}. \tag{4}$$

Thus,

$$D_k \quad = \quad \frac{c}{4\pi f_c} \cdot 10^{\frac{L_k - \eta_{LoS,k}}{20}}. \tag{5}$$

Let λ be the required time for the UAV to collect data from an IoT device. Let $R_k = \sqrt{{D_k}^2 - h^2} - \lambda V$, $\forall k$. Due to that the values of P_k's are not identical, the values of R_k's are not the same. Since V is the speed of the UAV, λV is the maximum distance that the UAV could move away from the kth IoT device within λ time units. Then, the UAV successfully collects data from the kth ground IoT device if the the distance between the backward path of the UAV and the kth IoT device is no greater than R_k, $\forall k$. Our goal is to find an optimal differential curve γ^* with minimum length in $\mathcal{R}^3$ such that $\gamma^*(0) = (x_d, y_d, h)$, $\gamma^*(1) = (x_s, y_s, h)$, $\gamma_3^*(\tau) = h$, $\forall \tau \in [0, 1]$, and $d(\gamma^*, (x_k, y_k, 0)) \le D_k$, $\forall k \in \mathcal{K}$. Recall that for each fixed $\tau \in [0, 1]$, $\gamma_3^*(\tau)$ is the third component of the vector $\gamma^*(\tau)$.

For each curve γ in $\mathcal{R}^3$, let $\tilde{\gamma}$ be the orthogonal projection of curve γ onto the horizontal plane $\mathbf{E}_2 = \{(x, y, z) \in \mathcal{R}^3 | z = 0\}$. Specifically, if $\gamma(\tau) = \{(\gamma_1(\tau), \gamma_2(\tau), \gamma_3(\tau)) | \tau \in [0, 1]\}$, $\tilde{\gamma}(\tau) = \{(\gamma_1(\tau), \gamma_2(\tau), 0) | \tau \in [0, 1]\}$. Let $PC_1(\mathcal{R}^2)$ be the set of continuous and piecewise differential plane curves on the plane $\mathbf{E}_2$. To find out an optimal backward trajectory for the UAV, we formulate the following optimization problem.

$$\begin{aligned}
&\min_{\tilde{\gamma}\in PC_1(\mathcal{R}^2)} \int_0^1 \sqrt{\big(\frac{d\tilde{\gamma}_1(\tau)}{d\tau}\big)^2 + \big(\frac{d\tilde{\gamma}_2(\tau)}{d\tau}\big)^2}\, d\tau \\
&\text{subject to} \\
&\tilde{\gamma} = \{(\tilde{\gamma}_1(\tau), \tilde{\gamma}_2(\tau), 0) | \tau \in [0,1]\} \\
&d(\tilde{\gamma}, \mathbf{w}_k) \le R_k, \forall k \in \mathcal{K} \\
&(\tilde{\gamma}_1(0), \tilde{\gamma}_2(0)) = (x_d, y_d) \\
&(\tilde{\gamma}_1(1), \tilde{\gamma}_2(1)) = (x_s, y_s). \qquad (6)
\end{aligned}$$

We now elaborate on the above optimization problem. First, based on differential geometry, the object function is the length of the differential curve $\tilde{\gamma}$. The first constraint corresponds to a parameterization of the plane curve $\tilde{\gamma}$. The second constraint states that the backward path of the UAV has to be close enough to each ground IoT device k in order to successfully collect and decode data, $\forall k$. The third constraint specifies the beginning point of the backward path, which is also the termination point of the forward path. The fourth constraint specifies the termination point of the backward path of the UAV, which is the start point of the forward path.

3.2 Collision-free paths

Avoiding collisions among UAVs is essential for a UAV communication network. First, two or more UAVs on their forward paths should not collide. Second, two or more UAVs on their backward paths should not collide. Third, a UAV on its forward path should not collide with another UAV on its backward path. Since the collision avoidance problems in the above three cases are essentially identical, it is sufficient to concentrate on the collision avoidance problem for UAVs on their forward paths.

We now consider the forward paths of the M UAVs and focus on collision avoidance. Let $q_m(t) = (x_m(t), y_m(t), h)$ be the coordinates of the location of UAV m at time t, $\forall m \in \{1, 2, .., M\}, t \ge 0$. Recall that d_{min} is the minimum distance required for two UAVs to avoid collisions. To completely avoid collisions, the following constraint has to be satisfied.

$$\|q_m(t) - q_n(t)\| \ge d_{min}, \forall m \ne n, t \ge 0. \qquad (7)$$

When the distance between two UAVs is less than or equal to d_{min}, the two UAVs adjust their altitudes according to their indexes to avoid collisions. Recall that d_{sen} is the sensing radius of a UAV. A UAV is able to obtain the relative position and velocity of an adjacent UAV if and only if their distance is no greater than d_{sen}. It is assumed that a UAV observes its environment every t_{sensor} time units. To avoid collisions, the value of t_{sensor} cannot be too large. On the other hand, to reduce computational complexity and energy consumption, the value of t_{sensor} cannot be too small. To completely avoid collisions, it is required that $d_{sen} \ge 2V \cdot t_{sensor} + d_{min}$. Note that $2V \cdot t_{sensor}$ is the maximum reduction for distance between two UAVs within t_{sensor} time units. If the distance between two UAVs is $2V \cdot t_{sensor} + d_{min}$ at time t_0 and the two UAVs fly toward each other, the distance between the two UAVs will become d_{min} at time $t_0 + t_{sensor}$. Without loss of essential generality, it is assumed that $t_{sensor} = 1$.

Let T_m be the total time required for UAV m to fly from $q_{start,m}$ to $q_{goal,m}$, $\forall m$. Define $\mathbf{T} = (T_1, T_2, .., T_M)$. Let $q_m = \{q_m(\tau) | 0 \le \tau \le T_m\}$ be the forward path of UAV m, $\forall$. Define $\mathbf{q} = (q_1, q_2, .., q_M)$. To deliver goods as soon as possible and reduce the energy consumption due to flying, we formulate the following optimization problem.

$$\begin{aligned}
&\min_{\mathbf{T},\mathbf{q}} \max_{m:1\le m\le M} \int_0^{T_m} \sqrt{\big(\frac{dx_m(\tau)}{d\tau}\big)^2 + \big(\frac{dy_m(\tau)}{d\tau}\big)^2}\, d\tau \\
&\text{subject to} \\
&T_m > 0, \forall m \in \{1, 2, .., M\} \\
&q_m = \{q_m(\tau) = (x_m(\tau), y_m(\tau), h) | 0 \le \tau \le T_m\}, \\
&\qquad \forall m \in \{1, 2, .., M\} \\
&q_m \in PC_1(\mathcal{R}^3), \forall m \in \{1, 2, .., M\} \\
&q_m(0) = (x_s, y_s, h), \forall m \in \{1, 2, .., M\} \\
&q_m(T_m) = (x_d, y_d, h), \forall m \in \{1, 2, .., M\} \\
&\left\| \frac{dq_m(t)}{dt} \right\| = V, \forall m \in \{1, 2, .., M\}, t \ge 0 \\
&\|q_i(t) - q_j(t)\| \ge d_{min}, \forall i \ne j, \forall t \ge 0 \\
&\mathbf{T} = (T_1, T_2, .., T_M) \\
&\mathbf{q} = (q_1, q_2, .., q_M). \qquad (8)
\end{aligned}$$

We now elaborate on the above optimization problem. First, $\int_0^{T_m} \sqrt{\big(\frac{dx_m(\tau)}{d\tau}\big)^2 + \big(\frac{dy_m(\tau)}{d\tau}\big)^2}\, d\tau$ is the forward trajectory length of UAV m. To minimize the completion time of delivering goods and reduce the overall energy consumption of UAV flights, we aim to minimize the maximum length over all forward paths of UAVs. The first constraint reflects that T_m is a positive real number. The second constraint defines the curve q_m, which is the forward path of UAV m, $\forall m$. The third constraint requires that q_m is a differential curve, $\forall m$. Therefore, $q_m(\tau)$ is also a continuous function of τ. The fourth constraint specifies the start point of curve q_m, while the fifth constraint specifies the termination point of curve q_m. The sixth constraint is due to that the speed of a UAV does not change with time and is equal to V. The seventh constraint is used to avoid collisions between two UAVs.

3.3 On the joint optimization problem

Ideally, the collision avoidance problem and the trajectory planning problem should be addressed under a joint optimization problem. In principle, the joint optimization problem could be formulated as a non-cooperative multi-agent sequential decision problem [35] [36]. However, the number of strategy profiles for multiple UAVs is expected to be very large. In addition, since the sensing radius of a UAV is finite, the joint optimization problem is a partially observable game or an imperfect information game [35]. To reduce the computational complexity and to benefit from the mathematical structure of the trajectory planning problem, we propose using the planned trajectory as the starting point for the learning-based collision avoidance algorithm. As shown later in the paper, the proposed approach could avoid collisions and produce shorter trajectories with lower computational complexity. In Section 4, we propose efficient algorithms for obtaining planned trajectories. In Section 5, we propose a decentralized approach based on reinforcement learning for UAV collision avoidance.

4 Trajectory Planning for Data Collection

It is difficult to directly solve (6), which looks for an optimal curve among all differential curves in the plane. In this section, we propose the convex-TSP algorithm to obtain an optimal backward path for a UAV among all curves that are composed of a finite number of line segments. The proposed algorithm is based on solving convex optimization problems and an auxiliary no-return traveling salesman problem (TSP).

The proposed algorithm consists of three parts. In the first part, it solves an auxiliary no-return TSP for determining the visiting order of ground IoT devices associated with the UAV. In the second part, given the visiting order, the proposed algorithm solves convex optimization problems to obtain potential turning points for the UAV. The potential turning points could be used to form line segments of a curve that is a feasible solution of (6). In the third part, the proposed algorithm comes up with the final solution based on refining the curve obtained in the second part. Pseudo codes for the proposed algorithm are included in Algorithm 1. The output of the proposed convex-TSP algorithm is a curve that consists of line segments.

4.1 No-return TSP for determining the visiting order

We adopt an auxiliary TSP based on the following observations. First, when $R_k = 0$, $\forall k$, the optimization problem (6) is very similar to the well-known Euclidean Traveling Salesman Problem (TSP) except that the former has different start point and termination point. Namely, we aim to solve a no-return TSP for determining the visiting order of ground IoT devices. In addition, when $R_k > 0$, $\forall k$, the optimization problem (6) is similar but not identical to the TSPN. While the Euclidean TSP and the TSPN are both NP-hard [6] [14] [37] [38], there exist efficient approximation algorithms and modern software tools for solving them. Therefore, we propose using an auxiliary TSP to determine an optimal visiting order for a given set of ground IoT devices in a systematic manner.

We now create an auxiliary TSP as follows. First, there are $K+3$ cities in the TSP. Specifically, city k corresponds to ground IoT device k, $\forall k \in \{1,2,..,K\}$, city $K+1$ corresponds to point S at (x_d, y_d), city $K+2$ corresponds to point D at (x_s, y_s), and city $K+3$ is the unique virtual city. City k is called a real city, $\forall k \in \{1,2,..,K+2\}$. Define $\mathbf{w}_{K+1} = (x_d, y_d)$ and $\mathbf{w}_{K+2} = (x_s, y_s)$. In a feasible solution of the auxiliary TSP, the traveler starts from the virtual city, visits $K+2$ real cities, and then goes back to the virtual city. The traveler visits each real city once. Since we try to determine an optimal backward path, city $K+1$ at (x_d, y_d) should be the second visited city and city $K+2$ at (x_s, y_s) should be the second last visited city of the traveler.

	virtual city	S	D	A	B
virtual city $(K+3)$	0	0	0	∞	∞
S $(K+1)$	∞	0	$d(S,D)$	$d(S,A)$	$d(S,B)$
D $(K+2)$	0	$d(D,S)$	0	$d(D,A)$	$d(D,B)$
A $(\leq K)$	∞	∞	$d(A,D)$	0	$d(A,B)$
B $(\leq K)$	∞	∞	$d(B,D)$	$d(B,A)$	0

TABLE 1: The cost matrix $\mathbf{C}$ for the auxiliary TSP.

Let $\mathbf{C}$ be the $(K+3)$-by-$(K+3)$ cost matrix for the auxiliary TSP. We set values for the elements of $\mathbf{C}$ as follows. First, $[\mathbf{C}]_{i,i} = 0$, $\forall i$. Second, $[\mathbf{C}]_{i,j} = \|\mathbf{w}_i - \mathbf{w}_j\|$, $\forall i, j \in \{1,2,..,K\}$. Namely, if city i and city j correspond to IoT devices, $[\mathbf{C}]_{i,j}$ is equal to the distance between the two IoT devices. Third, $[\mathbf{C}]_{K+3,K+1} = [\mathbf{C}]_{K+3,K+2} = 0$, while $[\mathbf{C}]_{K+3,j} = \infty$, $\forall j \in \{1,2,..,K\}$. Namely, the distance from the virtual city to point S/D is set to zero, while the distance from the virtual city to a city corresponding to an IoT device is set to infinity. By doing so, the traveler always moves from the virtual city to S or D. To ensure that the traveler always moves to the virtual city via D, $[\mathbf{C}]_{i,K+3} = \infty$, $\forall i \in \{1,2,..,K+1\}$, and $[\mathbf{C}]_{K+2,K+3} = 0$. Last, for completeness, $[\mathbf{C}]_{i,K+1} = \infty$, $\forall i \in \{1,2,..,K\}$, $[\mathbf{C}]_{K+2,K+1} = \|(x_d, y_d) - (x_s, y_s)\|$, and $[\mathbf{C}]_{i,K+2} = \|(x_i, y_i) - (x_s, y_s)\|$, $\forall i \in \{1,2,..,K+1\}$.

In Table 1, we show the cost matrix $\mathbf{C}$ for the auxiliary TSP. In the table, an element in the first column of the table represents the row index of the cost matrix, while an element in the first row represents the column index of the cost matrix. In addition, A represents a city that corresponds to an IoT device. Similarly, B is a city that corresponds to an IoT device. For example, if A is city a and B is city b, then $[\mathbf{C}]_{a,b} = \|\mathbf{w}_a - \mathbf{w}_b\|$, which is dnoted by $d(A,B)$, $\forall a, b \in \{1,2,..,K\}$. Moreover, $[\mathbf{C}]_{K+3,K+1} = 0$, since the distance from the virtual city to the source point S is set to 0.

After solving this virtual-city-aided TSP by Google Optimization Tools [39], we remove the virtual city, the start point S and the destination D to obtain the visiting order of ground IoT devices. Let $\alpha(k)$ be the index of the kth IoT device to be visited by the UAV based on solving the auxiliary TSP. The following theorem assures the correctness of the proposed approach.

Theorem 1:

(1) In an optimal solution of the auxiliary TSP, the second visited city is city $K+1$ at point S and the second last visited city is city $K+2$ at point D.

(2) An optimal tour of the auxiliary TSP corresponds to an optimal solution of the no-return TSP.

Proof: See Appendix. ■

4.2 Convex optimization for obtaining optimal visiting points

Given $[\alpha(1), \alpha(2), .., \alpha(K)]$, a list that represents the visiting order for IoT devices obtained by solving the auxiliary TSP problem, the proposed convex-TSP algorithm solves convex optimization problems to obtain a backward path of the UAV.

We now consider the problem of obtaining an optimal point for the UAV to visit the neighborhood of IoT device $\alpha(k)$ when the UAV plans to fly from $(\mathbf{u}_1, h)$ to $(\mathbf{u}_2, h)$, where $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{R}^2$. There are three cases. First, if $\mathbf{u}_1 \in C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$, we select $\mathbf{u}_1$ to be the optimal visiting point for IoT device $\alpha(k)$ (in the plane). Second, if $\mathbf{u}_1 \notin C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$ but $\mathbf{u}_2 \in C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$, we select $\mathbf{u}_2$ to be the optimal visiting point for IoT device $\alpha(k)$. Third, if $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{R}^2 - C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$, we formulate the following optimization problem.

$$P_k(\mathbf{u}_1, \mathbf{u}_2): \quad \min_{\mathbf{v}} \|\mathbf{v} - \mathbf{u}_1\| + \|\mathbf{v} - \mathbf{u}_2\| \quad \text{subject to} \quad \mathbf{v} \in C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)}). \tag{9}$$

We now elaborate on the above optimization problem, denoted by $P_k(\mathbf{u}_1, \mathbf{u}_2)$. First, the object function is the sum length of line segment $\overline{\mathbf{u}_1\mathbf{v}}$ and line segment $\overline{\mathbf{v}\mathbf{u}_2}$. It is the length of the UAV trajectory from $\mathbf{u}_1$ to $\mathbf{u}_2$ via $\mathbf{v}$. The constraint reflects that $\mathbf{v}$ has to be close enough to IoT device $\alpha(k)$ for the UAV to successfully collect and decode data from IoT device $\alpha(k)$. It is known that vector norm is a convex function [40]. Thus, $\|\mathbf{v} - \mathbf{u}_1\|$ and $\|\mathbf{v} - \mathbf{u}_2\|$ are both convex functions of $\mathbf{v}$. In addition, the sum of two convex functions is a convex function. Thus, $\|\mathbf{v} - \mathbf{u}_1\| + \|\mathbf{v} - \mathbf{u}_2\|$ is a convex function of $\mathbf{v}$. Furthermore, since the set of feasible solutions $C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$ is a circle, it is a convex set. Therefore, the above optimization problem is a convex optimization problem. Since problem (9) is a convex optimization problem, we can solve it by CVXPY [41] [42], which is a Python-embedded modeling language for convex optimization problems.

Let $\mathbf{v}_k^\dagger(\mathbf{u}_1, \mathbf{u}_2)$ be an optimal solution of the above optimization problem $P_k(\mathbf{u}_1, \mathbf{u}_2)$. When it is clear from the context, $\mathbf{v}_k^\dagger(\mathbf{u}_1, \mathbf{u}_2)$ is abbreviated by $\mathbf{v}_k^\dagger$. Geometric descriptions based on an ellipse for $\mathbf{v}_k^\dagger$ can be found in [43] [6]. In particular, [43] contains a proof that normal bisects the angle between the lines to the foci of an ellipse. To benefit from theory and modern software of convex optimization, we take a convex optimization approach instead.

We have to consider two cases in order to obtain the value of $\mathbf{v}_k^\dagger$. If the line segment from $\mathbf{u}_1$ to $\mathbf{u}_2$ does not intersect with the communication circle of IoT device $\alpha(k)$, the optimal point for the UAV to visit the neighborhood of IoT device $\alpha(k)$ when it flies from $\mathbf{u}_1$ to $\mathbf{u}_2$ is on the circumference of the circle $C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$.

Lemma 1: For the optimization problem $P_k(\mathbf{u}_1, \mathbf{u}_2)$, if $\overline{\mathbf{u}_1\mathbf{u}_2} \cap C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)}) = \emptyset$, $\mathbf{v}_k^\dagger \in \partial C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)})$.

Proof: See Appendix. ■

On the other hand, if $\overline{\mathbf{u}_1\mathbf{u}_2} \cap C_{\mathbf{w}_{\alpha(k)}}(R_{\alpha(k)}) \neq \emptyset$, $\mathbf{v}_k^\dagger$ is equal to $\mathbf{w}_k$'s foot of perpendicular on $\overline{\mathbf{u}_1\mathbf{u}_2}$.

As an illustration for Lemma 1, in Figure 2, we show the optimal point p' for the UAV to visit the neighborhood of IoT device k when the UAV plans to fly from point S to point D. In particular, the optimal point p' has to be on the circumference of the circle centered at $\mathbf{w}_k$ with radius R_k.

The second part of the proposed convex-TSP algorithm contains a loop for solving K instances of the optimization problem (9). Specifically, the kth instance is used to obtain the visiting point for IoT device $\alpha(k)$, $\forall k \in \{1, 2, .., K\}$. In the first instance, $\mathbf{u}_1 = (x_d, y_d)$ and $\mathbf{u}_2 = (x_s, y_s)$. For each $k \geq 2$, in the kth instance, $\mathbf{u}_1$ is equal to $\mathbf{v}_{k-1}^\dagger$, the visiting point for IoT device $\alpha(k-1)$, while $\mathbf{u}_2 = (x_s, y_s)$. The backward path obtained in the second part of the convex-TSP algorithm is composed of line segments that connect (x_d, y_d), $\mathbf{v}_1^\dagger$, $\mathbf{v}_2^\dagger$,..,$\mathbf{v}_K^\dagger$, and (x_s, y_s).

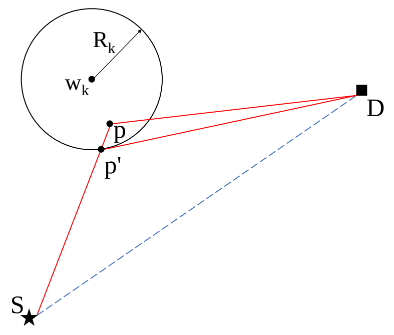
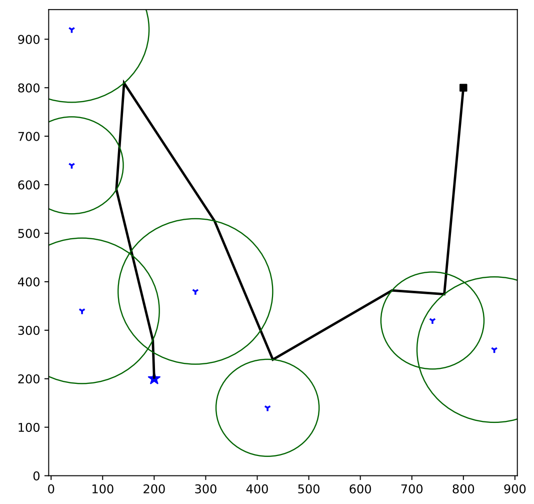


Fig. 2: The selection of an optimal visiting point.

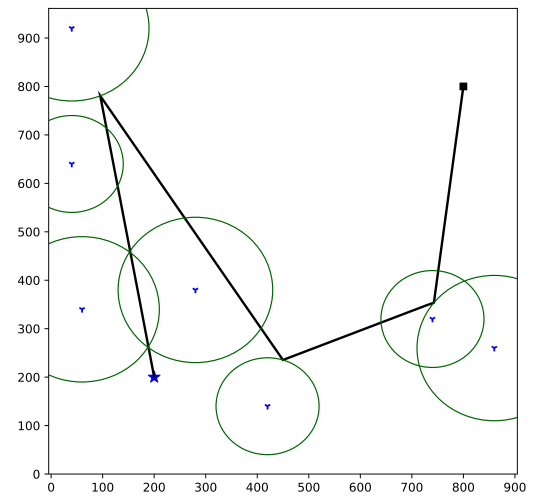
4.3 Refining the path

The backward path obtained in part 2 of the proposed algorithm is not necessarily optimal. It is mainly due to that the proposed algorithm obtains the value of $\mathbf{v}_k^\dagger$ without knowing the value of $\mathbf{v}_{k+1}^\dagger$. Thus, based on $(\mathbf{v}_1^\dagger, \mathbf{v}_2^\dagger, .., \mathbf{v}_K^\dagger)$, the third part of the proposed algorithm tries to get a better set of turning/bridging points $(\mathbf{v}_1^*, \mathbf{v}_2^*, .., \mathbf{v}_K^*)$ for the backward path of the UAV.

We now elaborate on the third part of the proposed convex-TSP algorithm. The third part is a path refinement subroutine. Pseudo codes for the third part of the proposed convex-TSP algorithm are included in Algorithm 2. The second part of the proposed convex-TSP algorithm uses $\mathcal{B} = [B_1, B_2, .., B_K]$, a list of bridging points on the plane $\mathbf{E}_2$, to represent a backward path composed of line segments. Namely, the backward path produced by the second part of the convex-TSP algorithm is composed of $\overline{SB_1}$, $\overline{B_1B_2}$,..,$\overline{B_{K-1}B_K}$, and $\overline{B_KD}$. The third part of the convex-TSP algorithm refines the backward path by changing the values of B_k's. In particular, the path refinement subroutine uses a while loop to find better solutions until it cannot obtain a solution that is significantly better than the previous one. In the pseudo codes, d represents the length of the best path in the previous iteration, d' represents the length of the best path in the current iteration, and δ is a predetermined positive real number. If the difference between d' and d is less than or equal to δ, Algorithm 2 terminates, since it fails to find out a path that is much better than the previous one. Inside the while loop, there exists a for loop. The for loop is responsible for adjusting the line segments of the best path up to date. As the second part, the third part of the proposed convex-TSP algorithm solves optimization problems in order to find a better path for the UAV to collect data from ground IoT devices. To obtain the value of $\mathbf{v}_k^\dagger$, the second part of the proposed algorithm sets $\mathbf{u}_2 = (x_s, y_s)$ in (9). In contrast, the third part of the proposed algorithm sets



(a) The path produced by the second part of the algorithm.



(b) The path produced by the third part of the algorithm.

Fig. 3: An illustration for the paths produced by the convex-TSP algorithm.

$\mathbf{u}_2 = B_{k+1}$ in (9) for obtaining the value of $\mathbf{v}_k^*$, $\forall k$. Initially, $B_{k+1} = \mathbf{v}_{k+1}^\dagger$.

In Figure 3(a), we show a path produced by the second part of the proposed convex-TSP algorithm for the seven IoT devices in the network. The communication region of an IoT device is represented by a green circle. The path consists of eight line segments, since there are seven IoT devices in the example. In Figure 3(b), we show the path produced by the third part of the proposed convex-TSP algorithm based on refining the path in Figure 3(a). After the refinement, the path becomes shorter and is composed of four line segments in this example.

In Figure 4, we show the average path length for different number of iterations in the proposed convex-TSP algorithm. We randomly create 100 network topologies to obtain the average path length. When the number of IoT devices is between 10 and 20, the proposed convex-TSP algorithm almost always converges within 20 iterations.

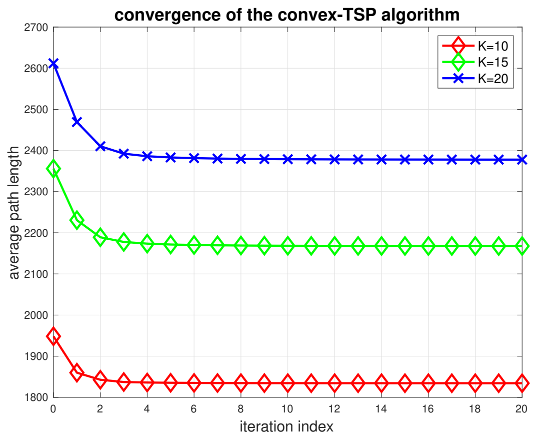


Fig. 4: The convergence of the proposed convex-TSP algorithm.

Algorithm 1 The convex-TSP algorithm for computing the backward path for a UAV.

Input: S (start point), D (destination), $\mathcal{K}$, $\{\mathbf{w}_k | k \in \mathcal{K}\}$, and δ.

Output: $\mathcal{B}$ (a list of bridging points that form the backward path).

```
1:  // Part 1: determine the visiting order for IoT devices.
2:  Obtain the visiting order for IoT devices by solving the Euclidean TSP for $\{\mathbf{w}_k | k \in \mathcal{K}\}$.
3:  Let $\alpha(k)$ be the index of the $k$th IoT device to be visited.
4:  Set $\mathcal{B} = [S, D]$ and $\mathcal{D} = \emptyset$. // $\mathcal{D}$ is a data store.
5:  // Part 2: determine the potential turning/visiting points.
6:  for $k = 1$ to $|\mathcal{K}|$ do
7:      Obtain $\mathbf{v}_{k,1}$, the optimal point on circumference of the circle centered at $\mathbf{w}_{\alpha(k)}$ with radius $R_{\alpha(k)}$.
8:      // $B_k$ is the $k$th point in the list $\mathcal{B}$, $\forall k \geq 1$.
9:      // $\overline{B_k B_{k+1}}$ denotes the line segment from $B_k$ to $B_{k+1}$.
10:     Obtain $\mathbf{v}_{k,2}$, the foot of perpendicular on $\overline{B_k B_{k+1}}$.
11:     if $d(\mathbf{w}_{\alpha(k)}, \overline{B_k B_{k+1}}) \leq R_{\alpha(k)}$ then
12:         $\mathbf{v}_k^\dagger \leftarrow \mathbf{v}_{k,2}$.
13:     else
14:         $\mathbf{v}_k^\dagger \leftarrow \mathbf{v}_{k,1}$.
15:     end if
16:     Insert $\mathbf{v}_k^\dagger$ to be between $B_k$ and $B_{k+1}$ in the list $\mathcal{B}$.
17:     Record $(\mathbf{v}_k^\dagger, \alpha(k))$ in $\mathcal{D}$.
18: end for
19: // Part 3: refine the path obtained in part 2.
20: Call Algorithm 2 with input $(\mathcal{B}, \mathcal{D})$ to refine the backward path.
```

Algorithm 2 The path refinement algorithm.

```
Input: $\mathcal{K}$, $\{\mathbf{w}_k | k \in \mathcal{K}\}$, $\mathcal{B}$, $\mathcal{D}$, $\delta$ and $r_{max}$.
Output: $\mathcal{B}$.
 1: $K \leftarrow |\mathcal{K}|$.
 2: Calculate $d' = \sum_{i=1}^{K+1} |\overline{B_{i-1}B_i}|$, the length of the path associated with $\mathcal{B}$.
 3: $d \leftarrow 0$, $r \leftarrow 0$.
 4: while $|d' - d| > \delta$ AND $r < r_{\max}$ do
 5:     for $k = 1$ to $K$ do
 6:         Obtain $\alpha(k)$ based on $\mathcal{D}$.
 7:         Let $\mathbf{v}_{k,1}$ be an optimal solution of (9) with $\mathbf{u}_1 = B_{k-1}$ and $\mathbf{u}_2 = B_{k+1}$.
 8:         Obtain $\mathbf{v}_{k,2}$, the foot of perpendicular on $\overline{B_{k-1}B_{k+1}}$.
 9:         if $d(\overline{B_{k-1}B_{k+1}}, \mathbf{w}_{\alpha(k)}) < R_{\alpha(k)}$ then
10:             $\mathbf{v}_k^* \leftarrow \mathbf{v}_{k,2}$.
11:         else
12:             $\mathbf{v}_k^* \leftarrow \mathbf{v}_{k,1}$.
13:         end if
14:         $B_k \leftarrow \mathbf{v}_k^*$.
15:         Record $(\mathbf{v}_k^*, \alpha(k))$ in $\mathcal{D}$.
16:     end for
17:     $d \leftarrow d'$.
18:     Calculate $d'$, the length of the path associated with $\mathcal{B}$.
19:     $r \leftarrow r + 1$.
20: end while
```

4.4 Computational complexity

We now analyze the computational complexity of the proposed convex-TSP algorithm. Let $O(T_c)$ be the computational complexity of solving the convex optimization problem (9). Let T_k be the computation complexity of the kth component of the proposed convex-TSP algorithm, $\forall k \in \{1, 2, 3\}$. Then, T_1 is the time required for solving the TSP associated with K IoT devices. Based on the Bellman-Held-Karp algorithm [44] [45], $T_1 \leq O(K^2 \cdot 2^K)$. In addition, there exists a polynomial-time 2-approximation algorithm for TSP with triangle inequality [38]. Recently, Traub *et al.* [46] proposed a polynomial-time algorithm with approximation guarantee $\frac{3}{2} + \epsilon$ for the s-t-path TSP, for any fixed $\epsilon > 0$. Based on Algorithm 1, T_2 is the time complexity of solving K different instances of the convex optimization problem (9). Then, $T_2 = O(K \cdot T_c)$. The computational complexity of the for loop in Algorithm 2 is $O(K \cdot T_c)$, since the for loop contains K convex optimization problems. The algorithm executes the while loop at most r_{max} times, where r_{max} is a predetermined positive integer. Therefore $T_3 = O(r_{max} \cdot K \cdot T_c)$. The overall computational complexity of the proposed algorithm is $O(T_1+T_2+T_3) \leq O(K^2 \cdot 2^K + (r_{max}+1) \cdot K \cdot T_c)$, when it obtains an exact solution for the auxiliary TSP. The proposed algorithm becomes a polynomial-time algorithm, when it uses a polynomial-time approximation algorithm [38] [46] for solving the auxiliary TSP.

5 Q-Learning for Collision-Free Navigation

In this paper, we propose using reinforcement learning for UAV collision avoidance. The proposed approach is distributed and a UAV does not know the paths/trajectories of other UAVs in advance. To formally introduce the proposed approach of reinforcement learning, we define the corresponding state space, action space, and reward function in this section. In addition, we show the adopted rule of value update.

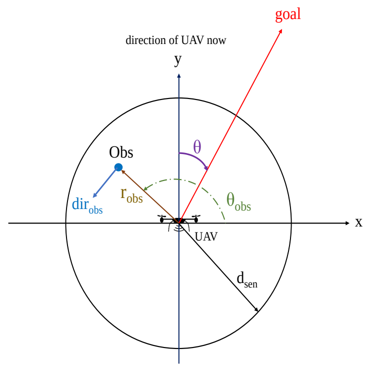


Fig. 5: The local state representation of a UAV for reinforcement learning.

Algorithm 3 The algorithm for setting the value of reward Φ_{t+1}.

```
Input: $(x_s, y_s)$, $(x_d, y_d)$, $V$, $(x_{old}, y_{old})$, $(x_{new}, y_{new})$, $\tilde{r}_{obs}$, and $d_{min}$.
Output: $\Phi_{t+1}$.
 1: $t_{direct} = \sqrt{(x_d - x_s)^2 + (y_d - y_s)^2}/V$.
 2: $d_{old} = \sqrt{(x_d - x_{old})^2 + (y_d - y_{old})^2}$.
 3: $d_{new} = \sqrt{(x_d - x_{new})^2 + (y_d - y_{new})^2}$.
 4: $t_{new} = d_{new}/V$.
 5: $\theta_{old} = \tan^{-1}(\frac{y_d - y_{old}}{x_d - x_{old}})$.
 6: $\theta_{new} = \tan^{-1}(\frac{y_d - y_{new}}{x_d - x_{new}})$.
 7: $\Delta\theta = \theta_{now} - \theta_{old}$.
 8: if $\tilde{r}_{obs} < d_{min}$ then
 9:     $\Phi_{t+1} = -0.333$.
10: else
11:     $\Phi_{t+1} = 1.03 \times \frac{d_{old} - d_{now}}{V}$.
12:     if $\Phi_{t+1} > 0$ then
13:         $\Phi_{t+1} \leftarrow \Phi_{t+1} \times (1 - \frac{t_{new}}{1.5 \cdot t_{direct}})$.
14:     else
15:         $\Phi_{t+1} \leftarrow \Phi_{t+1} \times (1 + \frac{t_{new}}{1.5 \cdot t_{direct}})$.
16:     end if
17:     $\Phi_{t+1} \leftarrow \Phi_{t+1} - |\Delta\theta|/180/6$.
18: end if
```

5.1 State space

Consider a tagged UAV. In Figure 5, we show the state representation from the viewpoint of the UAV. Specifically, the coordinates of the tagged UAV is always equal to $(0, 0)$. The y-axis corresponds to the current flying direction of the tagged UAV. In addition, θ is the angle between the direction to the destination point D and the y-axis. Suppose the tagged UAV detects an obstacle, which is a UAV that is at most d_{sen} away from the tagged UAV. Let r_{obs} be the distance between the tagged UAV and the detected obstacle.

Let θ_{obs} be the angle between the x-axis and the vector from the tagged UAV to the detected obstacle. Let dir_{obs} be the flying direction of the detected UAV. If the detected obstacle is static, dir_{obs} is set to a predetermined constant.

Let $\mathbf{x}_t$ be the state of the tagged UAV at (the beginning of) time slot t. $\mathbf{x}_t$ contains θ and information on detected obstacles. The information on a detected obstacle is denoted by info_{obs} and is based on quantizations of r_{obs}, θ_{obs}, and dir_{obs}. Let r_{scale}, θ_{scale}, and dir_{scale} be parameters for quantization. Specifically, $\text{info}_{obs} = (\lfloor \frac{r_{obs}}{r_{scale}} \rfloor + 1, \lfloor \frac{\theta_{obs}}{\theta_{scale}} \rfloor + 1, \lfloor \frac{\text{dir}_{obs}}{\text{dir}_{scale}} \rfloor + 1)$. For example, if $r_{obs} = 30$, $\theta_{obs} = 40$, $\text{dir}_{obs} = 50$, $r_{scale} = 5$, $\theta_{scale} = 10$, and $\text{dir}_{scale} = 10$, then $\lfloor r_{obs}/r_{scale} \rfloor + 1 = \lfloor 30/5 \rfloor + 1 = 7$, $\lfloor \theta_{obs}/\theta_{scale} \rfloor + 1 = \lfloor 40/10 \rfloor + 1 = 5$, and $\lfloor \text{dir}_{obs}/\text{dir}_{scale} \rfloor + 1 = \lfloor 50/10 \rfloor + 1 = 6$. In this case, $\text{info}_{obs} = (7, 5, 6)$.

To reduce the table size of reinforcement learning, we assume that a UAV takes into account at most two obstacles. Thus, $\mathbf{x}_t = [\theta, \text{info}_{obs1}, \text{info}_{obs2}]$, which represents a list containing three elements, θ, info_{obs1}, and info_{obs2}. Note that info_{obs1} is the information about obstacle 1, while info_{obs2} is the information about obstacle 2. If the tagged UAV detects three or more adjacent UAVs, the tagged UAV changes its altitude (based on the unique identification number) for collision avoidance.

The proposed state representation only contains relative position and flying direction between two UAVs. It does not contain the absolute positions of UAVs. Thus, the proposed reinforcement learning approach can be implemented in a distributed manner. It does not need any centralized controller that collects/knows the states of all UAVs.

5.2 Action space

Since it is assumed that a UAV can only change the flying direction once in a time slot, the path of a UAV consists of line segments. In addition, the speed of a UAV is constant. Thus, a_t is the turning angle of the UAV at the beginning of time slot t. Let θ_{max} be the maximum turning angle of a UAV in degree. To reduce the computational complexity, the action space $\mathcal{A}$ is composed of a finite number of real numbers in the set $[-\theta_{max}, \theta_{max}]$. For example, $\mathcal{A} = \{-45, -40, \ldots, 40, 45\}$. Note that $a_t \in \mathcal{A}$, $\forall t$. Let θ_t be the angle of flying of the tagged UAV in time slot t. Then, $\theta_t = (\theta_{t-1} + a_t) \mod 360$.

5.3 Reward and value update

When the tagged UAV takes action a_t at state $\mathbf{x}_t$ in time slot t, it gains rewards with amount equal to Φ_{t+1}. Let (x_{old}, y_{old}, h) be the location of the UAV in the current time slot. Let (x_{new}, y_{new}, h) be the location of the UAV in the next time slot after taking action a_t. There are three factors for determining the reward of a state-action pair. The first factor is the distance between the UAV and the obstacles, since it is essential to avoid collisions. Let $\tilde{r}_{obs}$ be the distance between the tagged UAV and an adjacent UAV after taking the action a_t at state $\mathbf{x}_t$. Recall that d_{min} is the minimum distance for two UAVs to avoid collisions. If $\tilde{r}_{obs} < d_{min}$, to avoid collisions, the reward Φ_{t+1} is set to a negative real number such as -0.333. The second factor is the distance between a UAV and its goal, since the ultimate goal of a UAV is to arrive at the goal as soon as possible. Let d_{old} be the distance between the UAV and its destination in the current time slot. Let d_{new} be the distance between the UAV and its destination in the next time slot if the action a_t is taken at state $\mathbf{x}_t$. The reward Φ_{t+1} is linearly proportional to $d_{old} - d_{new}$. The third factor is $\Delta\theta$, which is the angle between the flying direction of the UAV after taking the action a_t at state $\mathbf{x}_t$ and the direction towards its goal. Since it is desired to deliver the goods to the destination as early as possible, the smaller the absolute value of $\Delta\theta$ is, the larger the reward is. In Algorithm 3, we show the pseudo codes for setting the value of the reward Φ_{t+1}.

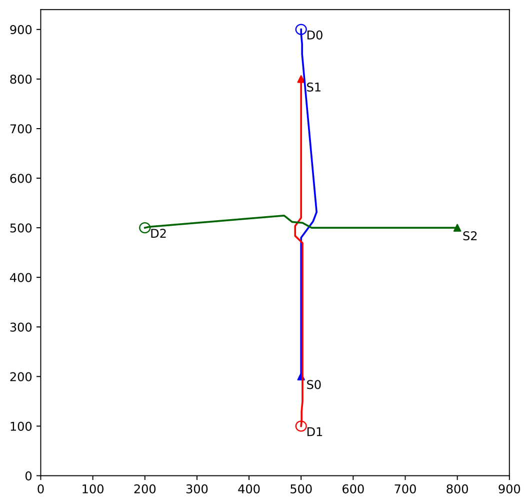


Fig. 6: An illustration of collision avoidance for 3 UAVs based on the proposed approach of reinforcement learning.

Let $\alpha \in (0, 1)$ be the learning rate and $\gamma \in (0, 1)$ be the reward discount factor. Let $Q(\mathbf{x}_t, a_t)$ be the expected value of the total discounted reward for state $\mathbf{x}_t$ and action a_t. According to (6.8) in [2], the value of the Q function is updated as follows.

$$\begin{aligned} Q(\mathbf{x}_t, a_t) \quad \leftarrow \quad & Q(\mathbf{x}_t, a_t) + \alpha\big[\Phi_{t+1} + \\ & \gamma \max_{a_{t+1}} Q(\mathbf{x}_{t+1}, a_{t+1}) - Q(\mathbf{x}_t, a_t)\big]. \end{aligned} \tag{10}$$

5.4 Validation of collision avoidance

To show that the proposed reinforcement learning approach allows the UAVs to successfully avoid collisions, we show UAV trajectories in Figure 6. Paths of different colors correspond to paths of different UAVs. Three UAVs are indexed by 0, 1, and 2, respectively. The source of UAV k is marked by Sk and the destination of UAV k is marked by Dk, $\forall k \in \{0, 1, 2\}$. In Figure 6, the three UAVs successfully avoid collisions. In particular, UAV 0 successfully escapes collisions from UAV 1 and UAV 2. In addition, for each UAV, the actual trajectory is very close to the shortest path from its source point to its destination point.

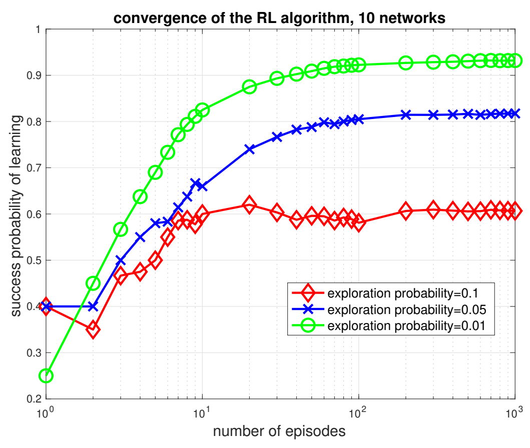


Fig. 7: An illustration for the success probability of reinforcement learning.

5.5 Convergence and computational complexity

In Figure 7, we show the convergence of the sequence of success probability of learning for the adopted reinforcement learning algorithm. To evaluate the reinforcement algorithm, we randomly create 10 networks. For each network, the reinforcement learning algorithm is trained for 1000 episodes. In an episode, whenever the reinforcement learning algorithm has to choose an action for a state, it chooses an optimal action up to date with probability $1-\delta$ or randomly chooses an action with probability δ. The value of δ is called the exploration probability. For an episode, the reinforcement learning algorithm is successful if all UAVs are able to deliver goods and collect data from ground IoT devices without collisions. The success probability for n episodes is the number of successful episodes divided by n. When $\delta \in \{0.01, 0.05, 0.1\}$, the success probability of learning converges within 1000 episodes. As the value of δ decreases from 0.1 to 0.01, the sequence of success probability of learning converges more slowly but the corresponding limit increases. When δ is smaller, the reinforcement learning algorithm adopts an optimal action in each state with a larger probability. Thus, once the reinforcement learning algorithm obtains a successful strategy, it will stick to the successful strategy with a larger probability. Therefore, the limit for the sequence of success probability is larger. For a network, it is sufficient to have a successful episode, since the corresponding Q-table will be used after the training phase.

We now analyze the computational complexity of the proposed distributed reinforcement learning approach. There are two phases for the distributed reinforcement learning approach. The first phase is the training phase, while the second phase is the deployment phase. Let $\mathcal{S}$ be the state space. Recall that $\mathcal{A}$ is the action space. We use a table and an array to implement the proposed distributed reinforcement learning approach. Specifically, we use a table to store the values of $Q(s,a)$'s such that each row corresponds to a state in the state space $\mathcal{S}$ and each column corresponds to an action in $\mathcal{A}$. In addition, the states in the table are sorted. Furthermore, we adopt an array where the kth element corresponds to the optimal action for the kth state in the table. Thus, the space complexity for the learning phase of the proposed distributed reinforcement learning approach is $O(|\mathcal{S}|\cdot|\mathcal{A}|)+O(|\mathcal{S}|)=O(|\mathcal{S}|\cdot|\mathcal{A}|)$. On the other hand, in the deployment phase, only the array is required. Thus, the space complexity of the deployment phase is $O(|\mathcal{S}|)$. Typically, the time complexity of the learning phase is linearly proportional to the product of the number of episodes and the number of time slots per episode. Since the learning phase is done offline before the deployment phase, we focus on the time complexity of the deployment phase. When binary search is used, it takes $O(\log_2|\mathcal{S}|)$ time to find the current state from the table/array. Once the location of the current state in the table/array is obtained, it takes $O(1)$ time to obtain the optimal action for the state based on the array. Therefore, the time complexity for the deployment phase is $O(\log_2|\mathcal{S}|)\times O(1)=O(\log_2|\mathcal{S}|)$.

6 Simulation Setup and Results

In this section, we show simulation results that justify the usage of the proposed approach. We wrote Python programs to obtain simulation results. In addition, we use the Google Optimization tools [39] to solve the traveling salesman problems. Furthermore, we use CVXPY [41] [42] to solve convex optimization problems. The mission area for UAVs to deliver goods and collect data from ground IoT devices is a square of size 1 km $\times$ 1 km. In Table 2, we show the values of key parameters in the simulation. In addition, we set $\delta=0.1$ in Algorithm 2.

Parameter	Description	Value
P_k	transmit power of device k	14 or 16 mW
P_n	noise power	-130 dBm
Γ_k	received SNR threshold of device k	50 dB
h	UAV's altitude	300 or 100 (m)
η_{LoS}	average additional loss for LoS	3 dB
f_c	carrier frequency	2 GHz
V	the speed of a UAV	20 m/s
λ	required communication time	0.1 s
d_{min}	minimum distance between UAVs	5 m
d_{sen}	radius of UAV's sensing area	45 m

TABLE 2: Key parameters in the simulation.

6.1 Backward trajectories

We compare our proposed algorithm with three algorithms of path planning: the devices as waypoints algorithm, the random geometric disk cover (GDC) algorithm [23], and the greedy algorithm.

The devices as waypoints algorithm works as follows. First, it uses the locations of ground IoT devices as waypoints. Second, given the waypoints, it solves the auxiliary TSP to obtain an order for visiting the waypoints. Two waypoints are said to be adjacent, if there exists an integer $k\geq 1$ such that one of them is the kth visited waypoint and the other is the $(k+1)$th visited waypoint. Last, it uses the line segments connecting adjacent waypoints to form the UAV trajectory for data collection.

The random GDC algorithm works as follows. First, it generates waypoints by solving the GDC problem. Given the waypoints, it solves the auxiliary TSP to obtain the visiting order for the waypoints. Last, it uses the line segments connecting adjacent waypoints to form the UAV trajectory for data collection. We now elaborate on the procedure the random GDC algorithm uses to create the waypoints. Initially, all IoT devices are uncovered. For each uncovered IoT device k, it generates N_g circles with radius R_k. In addition, for each of the N_g circles that are associated with IoT device k, the distance between the center and $\mathbf{w}_k$ is a random variable that is uniformly distributed over the interval $[0, R_k]$. Next, it selects the optimal circle that covers the maximum number of uncovered IoT devices. The center of the selected circle becomes a waypoint and the IoT devices covered by the selected circle become covered. It repeats the above process until all IoT devices are covered.

Unlike the other three studied algorithms, the greedy algorithm does not obtain the visiting order of IoT devices based on TSP. Instead, in each iteration, it first identifies the IoT device that is furthest from D among the unvisited IoT devices. Consider an iteration. Let k_1 be the index of the IoT that is furthest from D in the iteration. There are three cases. First, if there is no IoT device with index $k_2 \neq k_1$ such that $d(\mathbf{w}_{k_1}, \mathbf{w}_{k_2}) \leq R_{k_1} + R_{k_2}$, the greedy algorithm selects $\mathbf{w}_{k_1}$ as the next visiting point. Second, if there exists a unique IoT device with index $k_2 \neq k_1$ such that $d(\mathbf{w}_{k_1}, \mathbf{w}_{k_2}) \leq R_{k_1} + R_{k_2}$, the greedy algorithm selects the point that is closest to the UAV among the set $C_{\mathbf{w}_{k_1}}(R_{k_1}) \cap C_{\mathbf{w}_{k_2}}(R_{k_2})$ as the next visiting point. In this case, when the projected location of the UAV is in the set $C_{\mathbf{w}_{k_1}}(R_{k_1}) \cap C_{\mathbf{w}_{k_2}}(R_{k_2})$, the UAV simultaneously covers the two IoT devices with indexes k_1 and k_2. Last, if more than two IoT devices can be concurrently covered, the greedy algorithm calculates the closest points for all pairs of the IoT devices as candidates. Then, among the candidates, it chooses the one that is closest to the UAV as the next visiting point.

We first study a network in which $h = 300$ and 10 ground IoT devices are deployed within a 1000 m $\times$ 1000 m square. In addition, there is a UAV that aims to fly from point S at $(200, 200)$ to point D at $(800, 800)$. Furthermore, the values of R_k's are randomly created. In Figure 8(a), we show the backward path produced by the proposed convex-TSP algorithm in a network. In Figure 8(b), we show the backward path produced by the random GDC algorithm in the same network. As shown in the figures, the visiting order produced by the random GDC algorithm for IoT devices is similar to that produced by the proposed convex-TSP algorithm. However, the features of the turning points produced by the proposed convex-TSP algorithm are very different from those produced by the random GDC algorithm. Although the turning points for the path produced by the proposed convex-TSP algorithm are on the boundaries of circles, a number of turning points for the path produced by the random GDC algorithm are not on the circumferences of any circles. More important, in this network topology, the proposed convex-TSP algorithm is superior to the random GDC algorithm.

In Figure 8(c), we show the backward path produced by the greedy algorithm in the same network. The visiting order produced by the greedy algorithm for IoT devices is very different from that produced by the proposed convex-TSP algorithm. The former does not rely on the TSP while the latter is based on the TSP. In this network topology, the proposed convex-TSP algorithm outperforms the greedy algorithm.

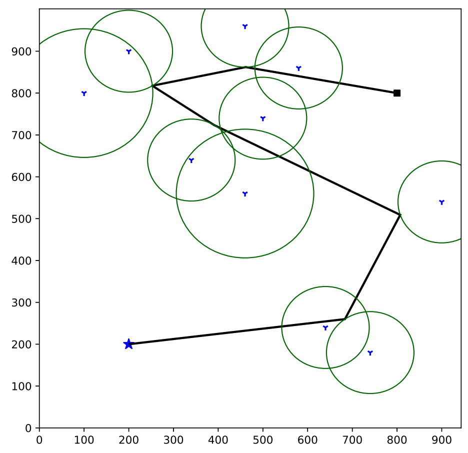
In Figure 9(a), we show the trajectory length as a function of number of ground IoT devices for the four studied schemes of path planning. In terms of the average trajectory length, the proposed convex-TSP algorithm outperforms the other three studied algorithms. On the other hand, the greedy algorithm is the worst. When there are 5 ground IoT devices, the greedy algorithm is superior to the random GDC algorithm. In contrast, when there are 15 or 20 ground IoT devices, the random GDC algorithm is better than the greedy algorithm.

In Figure 9(b), we show the average distance between the UAV trajectory to ground IoT devices, when $K = 15$. For the devices as waypoints algorithm, the average distance is always zero. It is due to that the algorithm selects the locations of ground IoT devices as waypoints. The proposed convex-TSP algorithm reduces the trajectory length at the cost of increasing the average distance between the UAV trajectory and associated ground IoT devices.

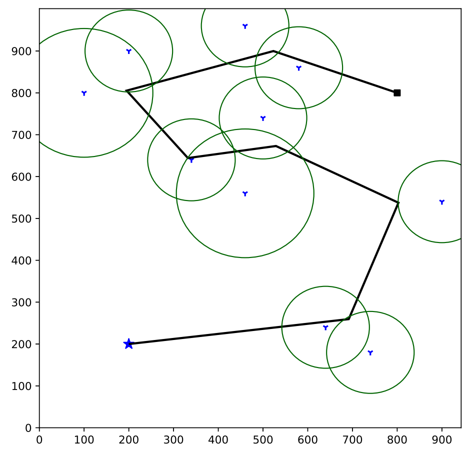
In addition to the above four algorithms, we also evaluate the circumcircle-based algorithm and the uniform disk algorithm. The circumcircle-based algorithm is a variant of the CSS algorithm in [3]. The CSS algorithm is designed for wireless sensors with the same communication radius and adopts circumcircles to reduce the path length. Let $R_{min} = \min_{i:1 \leq i \leq K} R_k$. The uniform disk algorithm is identical to the convex-TSP algorithm except that the former pretends that $R_i = R_{min}$, $\forall i$.

Let $\overline{R} > 0$ be the maximum communication radius for ground IoT devices. We study the case in which $h = 100$ and R_k is a random variable uniformly distributed over the set $\{50, 100, .., \overline{R}\}$, $\forall k$. In Figure 10, we show the average path length as a function of the maximum communication radius. As long as $\overline{R} \geq 100$, the proposed convex-TSP algorithm is the best among the six studied algorithms in terms of the average trajectory length. Except for the devices as waypoints algorithm, for each of the other studied algorithms, as the maximum communication radius increases, the average trajectory length decreases. As the maximum communication radius increases, the UAV has more freedom to choose its trajectory and therefore is able to select a shorter trajectory. When the ground IoT devices have distinct communication radii, the convex-TSP algorithm is superior to the circumcircle-based algorithm. In this case, the proposed convex-TSP algorithm adopts convex optimization to minimize the UAV path length. On the other hand, when all ground IoT devices have the same communication radius, the circumcircle-based algorithm slightly outperforms the convex-TSP algorithm, since the former uses circumcircles to efficiently reduce the number of turning points of a UAV trajectory. For the uniform disk algorithm, as the value of $\overline{R}$ increases, the probability that the minimum among realized radii of IoT devices is greater than 50 increases. Thus, as the value of $\overline{R}$ increases, the average trajectory length decreases slightly.

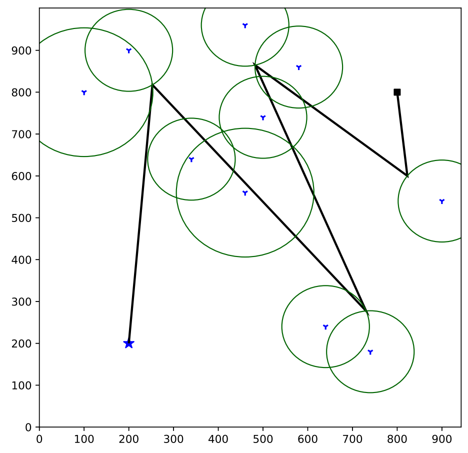
In Figure 11, we show the impact of the number of UAVs on the average trajectory length per UAV. We randomly create 100 networks and each network contains $K = 100$ ground IoT devices. The number of UAVs is between 2 and 10. In addition, the communication radius of a ground IoT device is a random variable uniformly distributed over the



(a) A trajectory produced by the convex-TSP algorithm.

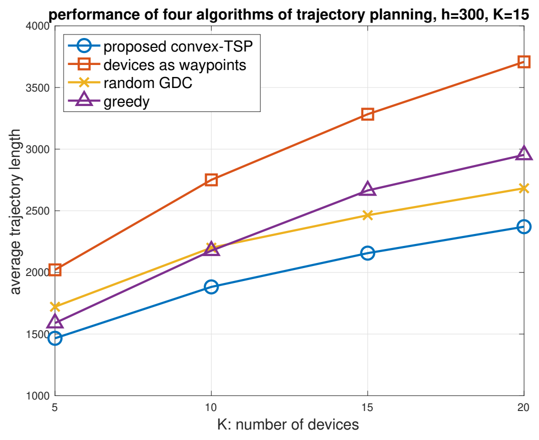


(b) A trajectory produced by the random GDC algorithm.

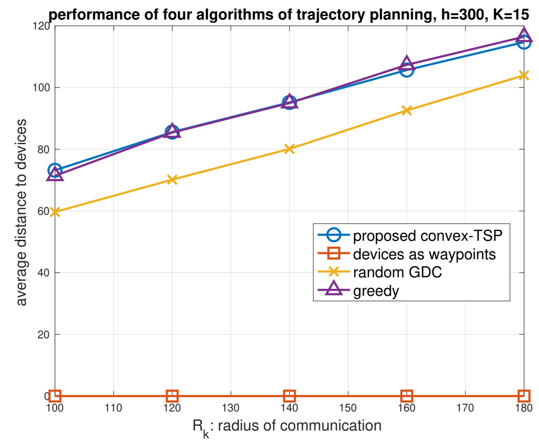


(c) A trajectory produced by the greedy algorithm.

Fig. 8: An illustration for the backward paths produced by three studied algorithms.



(a) The average trajectory length as a function of the number of IoT devices.



(b) The average distance between trajectory and devices as a function of radius of communication, when $K = 15$.

Fig. 9: Performance of four algorithms for trajectory planning, when $h = 300$.

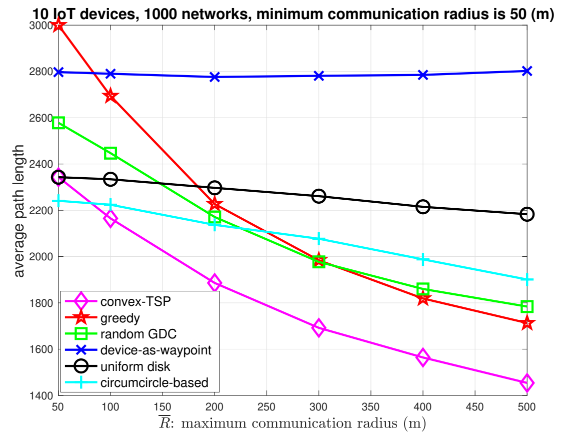


Fig. 10: The average trajectory length as a function of the maximum communication radius, when $h = 100$.

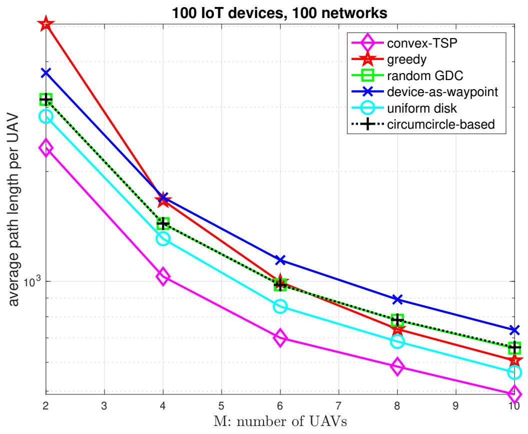


Fig. 11: The average trajectory length per UAV as a function of number of UAVs.

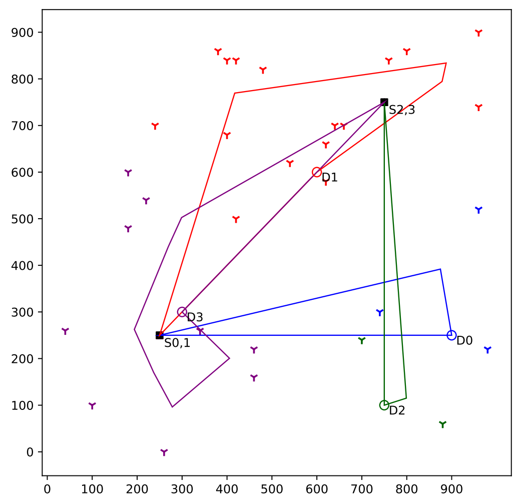
set $\{50, 100, 150, 200, 250, 300\}$. For each studied algorithm, the average trajectory length per UAV decreases as the value of M increases. As the value of M increases, a UAV tends to serve fewer ground IoT devices and therefore uses a shorter path. Among the studied algorithms, the proposed convex-TSP algorithm is the best regardless of the value of M. The convex-TSP algorithm adopts both combinatorial optimization and convex optimization to minimize the average trajectory length per UAV. When M is between 2 and 10, in comparison with the random GDC algorithm or the circumcircle-based algorithm, the proposed convex-TSP algorithm could reduce the average trajectory length per UAV by at least 25%. The average trajectory length per UAV of the circumcircle-based algorithm is almost identical to that of the random GDC algorithm. We now elaborate on the result. To cover all ground IoT devices by as few turning points as possible, the former adopts circumcircles to deterministically choose turning points, while the latter uses enough random circles to select turning points. When there are 100 IoT devices and no greater than 10 UAVs in the network, the two algorithms choose similar sets of turning points. Therefore, the random GDC algorithm has similar performance to the circumcircle-based algorithm.

	UAV 0	UAV 1	UAV 2	UAV 3
length of the planned trajectory	1434.975	1896.011	1338.174	1909.327
length of the actual trajectory	1473.536	1931.823	1350.598	1931.844

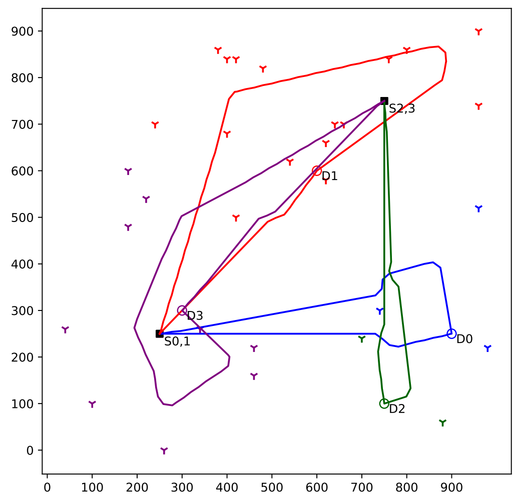
TABLE 3: The planned/actual trajectory length for each UAV in terms of meters.

6.2 Complete UAV trajectories

We also study the forward trajectories and backward trajectories together for UAVs. In the studied network, there are $M = 4$ UAVs and $N = 30$ ground IoT devices. An IoT device is associated with the UAV whose destination point is closest to it. The common distribution center of UAV 0 and UAV 1 is located at $(250, 250)$. The common distribution center of UAV2 and UAV3 is located at $(750, 750)$. The



(a) The planned trajectories for 4 UAVs.



(b) The actual trajectories for 4 UAVs.

Fig. 12: Complete trajectories for four UAVs.

delivery points for UAV 0, UAV 1, UAV 2, and UAV 3 are located at $(900, 250)$, $(600, 600)$, $(750, 100)$, and $(300, 300)$, respectively. The simulation is composed of two phases. The first phase is the learning phase in which UAVs fly and build up their own Q-tables based on the observations and the proposed reinforcement learning approach. In the second phase, the UAVs deliver goods and collect data from ground IoT devices. In addition, each UAV avoids collisions based on the Q-tables produced in the first phase.

In Figure 12(a), we show the planned trajectories for the four UAVs. The planned trajectories are ideal and their design does not take into account collision avoidance and the maximum magnitude of the turning angle in a time slot. Since a UAV could use the planned path to collect data from associated ground IoT devices, the planned path serves as the basis for the actual path.

In Figure 12(b), we show the real trajectories for the four UAVs. Each UAV follows the planned trajectory to a large extent but deviates from the planned trajectory when obstacles are detected. In addition, due to the constraint on the turning angle, a real trajectory is smoother than the corresponding planned trajectory. In Table 3, we show the lengths of the planned trajectories and the actual trajectories. A real UAV trajectory is slightly longer than the associated planned UAV trajectory as expected.

7 Conclusion

We have proposed novel approaches of collision avoidance and trajectory planning for UAV communication networks. Specifically, each UAV is responsible for delivering objects in the forward path and collecting data from heterogeneous ground IoT devices in the backward path. We have adopted reinforcement learning for assisting UAVs to avoid collisions without knowing the trajectories of other UAVs in advance. In addition, for each UAV, we have used combinatorial optimization and convex optimization to obtain an optimal path that assures data collection from all associated IoT devices. In particular, to obtain an optimal visiting order for IoT devices, we have formulated and solved a no-return traveling salesman problem. Given a visiting order, we have formulated and solved convex optimization problems to obtain line segments of an optimal backward path for heterogeneous ground IoT devices. We have demonstrated that the proposed reinforcement learning approach allows UAVs to successfully avoid collisions. In addition, we have used simulation results to show that the proposed approach is superior to a number of alternative approaches in terms of backward trajectory length. Future works include using deep Q-learning to further reduce the table size of reinforcement learning for efficient collision avoidance. An important direction of future research is to jointly optimize the forward path and the backward path for UAV-based goods delivery and IoT data collection, when the forward path does not have to be the shortest. Another important direction of future research is the joint optimization of collision avoidance and trajectory planning.

References

[1] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36-42, May 2016.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, 2nd Edition. The MIT Press, 2018.

[3] L. He, J. Pan, and J. Xu, "A Progressive Approach to Reducing Data Collection Latency in Wireless Sensor Networks with Mobile Elements," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1308-1320, July 2013.

[4] D. Kim, R. N. Uma, B. H. Abay, W. Wu, W. Wang, and A. O. Tokuta, "Minimum latency multiple data MULE trajectory planning in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 838–851, Apr. 2014.

[5] D. Kim, L. Xue, D. Li, Y. Zhu, W. Wang, and A. O. Tokuta, "On Theoretical Trajectory Planning of Multiple Drones To Minimize Latency in Search-and-Reconnaissance Operations," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3156-3166, Nov. 2017.

[6] Z. Chen, X. Zhu, X. Gao, F. Wu, J. Gu, and G. Chen, "Efficient scheduling strategies for mobile sensors in sweep coverage problem," in *Proc. 2016 IEEE SECON*, Jun. 2016, pp. 1-4.

[7] Y. Zeng, X. Xu, and R Zhang, "Trajectory Design for Completion Time Minimization in UAV-Enabled Multicasting," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2233-2246, Apr. 2018.

[8] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3179-3192, Nov. 2017.

[9] I. Mahjri, A. Dhraief, A. Belghith, and A. S. AlMogren, "SLIDE: A Straight Line Conflict Detection and Alerting Algorithm for Multiple Unmanned Aerial Vehicles," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1190-1203, May 2018.

[10] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124-2136, Mar. 2019.

[11] P.-Y. Lin, H.-T. Chiu, and R.-H. Gau, "Machine Learning-Driven Optimal Proactive Edge Caching in Wireless Small Cell Networks," in *Proc. 2019 IEEE VTC-Spring*, Kuala Lumpur, Malaysia, April 28-May 1, 2019.

[12] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile Unmanned Aerial Vehicles (UAVs) for Energy-Efficient Internet of Things Communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574-7589, Nov. 2017.

[13] D. Yang, Q. Wu, Y. Zeng, and R. Zhang, "Energy Tradeoff in Ground-to-UAV Communication via Trajectory Design," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6721-6726, Jul. 2018.

[14] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 9, pp. 1252-1261, Sep. 2007.

[15] Y. Li, G. Feng, M. Ghasemiahmadi, and L. Cai, "Power Allocation and 3-D Placement for Floating Relay Supporting Indoor Communications," *IEEE Trans. Mobile Comput.*, vol. 18, no. 3, pp. 618-631, Mar. 2019.

[16] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When Mobile Crowd Sensing Meets UAV: Energy-Efficient Task Assignment and Route Planning," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5526-5538, Nov. 2018.

[17] Q. Wu, Y. Zeng, and R. Zhang, "Joint Trajectory and Communication Design for Multi-UAV Enabled Wireless Networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109-2121, Mar. 2018.

[18] F. Cheng, S. Zhang, Z. Li, Y. Chen, N. Zhao, F. R. Yu, and V. C. M. Leung, "UAV Trajectory Optimization for Data Offloading at the Edge of Multiple Cells," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6732-6736, Jul. 2018.

[19] S. Zhang, Y. Zeng, and R. Zhang, "Cellular-Enabled UAV Communication: A Connectivity-Constrained Trajectory Optimization Perspective," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2580-2604, Mar. 2019.

[20] Nan Zhao, X. Pang, Z. Li, Y. Chen, F. Li, Z. Ding, and M.-S. Alouini, "Joint Trajectory and Precoding Optimization for UAV-Assisted NOMA Networks," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3723-3735, May 2019.

[21] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of The ACM*, vol. 32, no. 1, pp. 130-136, Jan. 1985.

[22] T. Erlebach and E. J. van Leeuwen, "Approximating geometric coverage problems," in *Proc. 2008 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1267-1276, Jan. 2008.

[23] Random geometric disk cover (GDC): http://jsfiddle.net/nwvao72r/4/

[24] R.-H. Gau and Y.-Y. Peng, "A Dual Approach for The Worst-Case-Coverage Deployment Problem in Ad-Hoc Wireless Sensor Networks," in *Proc. 2006 IEEE MASS*.

[25] Z. Zhang, J. Willson, Z. Lu, W. Wu, X. Zhu, and D.-Z. Du, "Approximating Maximum Lifetime k-Coverage Through Minimizing Weighted k-Cover in Homogeneous Wireless Sensor Networks," in *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3620-3633, Dec. 2016.

[26] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of UAV-mounted mobile base stations," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 604-607, Mar. 2017.

[27] E. Koyuncu, M. Shabanighazikelayeh, and H. Seferoglu, "Deployment and Trajectory Optimization of UAVs: A Quantization Theory Approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8531-8546, Dec. 2018.

[28] X. Zhang and L. Duan, "Fast Deployment of UAV Networks for Optimal Wireless Coverage," *IEEE Trans. Mobile Comput.*, vol. 18, no. 3, pp. 588-601, Mar. 2019.

[29] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747-3760, Jun. 2017.

[30] K. Li, W. Ni, X. Wang, R. P. Liu, S. S. Kanhere, and S. Jha, "Energy-Efficient Cooperative Relaying for Unmanned Aerial Vehicles," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1377-1386, June 2016.

[31] J. Xu, Y. Zeng, and R. Zhang, "UAV-enabled wireless power transfer: Trajectory design and energy optimization," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5092-5106, Aug. 2018.

[32] S. Yin, Y. Zhao, and L. Li, "Resource Allocation and Basestation Placement in Cellular Networks With Wireless Powered UAVs," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 1050-1055, Jan. 2019.

[33] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and Robert Schober, "Optimal 3D-Trajectory Design and Resource Allocation for Solar-Powered UAV Communication Systems," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281-4298, June 2019.

[34] M. Alzenad, A. El-Keyi, and H. Yanikomeroglu, "3-D Placement of an Unmanned Aerial Vehicle Base Station for Maximum Coverage of Users With Different QoS Requirements," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 38-41, Feb. 2018.

[35] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2nd Edition. Cambridge University Press, 2017.

[36] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition. Pearson, 2020.

[37] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[38] T. H. Cormen, C. E. Leiserson, R. L. Riverson, and C. Stein, *Introduction to Algorithms*, 3rd Edition. The MIT Press, 2009.

[39] Google Optimization Tools: https://developers.google.com/optimization/introduction/python

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press. 2004.

[41] S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1-5, 2016,

[42] A. Agrawal, R. Verschueren, S. Diamond and S. Boyd, "A Rewriting System for Convex Optimization Problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42-60, 2018.

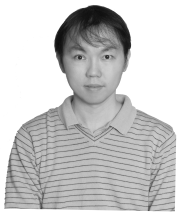
[43] https://en.wikipedia.org/wiki/Ellipse

[44] Richard Bellman, "Dynamic programming treatment of the traveling salesman problem," *Journal of The ACM*, vol. 9, no. 1, pp. 61-63, Jan. 1962.

[45] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal for the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196-210, 1962

[46] V. Traub and J. Vygen, "Approaching $\frac{3}{2}$ for the s-t-path TSP," *Journal of The ACM*, vol. 66, no. 2, Apr. 2019.

Yu-Hsin Hsu received the B.S. degree in electrical engineering and the M.S. degree in communications engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2017 and 2019, respectively. Her research interests include optimization for UAV wireless communication networks and machine learning for communication networks.



Rung-Hung Gau received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, the M.S. degree in electrical engineering from University of California at Los Angeles, Los Angeles, CA, USA, and the Ph.D. degree in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 1994, 1997, and 2001, respectively. He is currently a professor and the director of the Institute of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan. His current research interests include optimal resource allocation in NOMA wireless networks, machine learning and optimization for wireless communications and mobile computing, UAV communication networks, Internet of things, and software defined networking.