UAV Trajectory Planning for AoI-Minimal Data Collection in UAV-Aided IoT Networks by Transformer

Botao Zhu, Ebrahim Bedeer, *Member, IEEE*, Ha H. Nguyen, *Senior Member, IEEE*, Robert Barton, *Member, IEEE*, and Zhen Gao, *Member, IEEE*

Abstract—Maintaining freshness of data collection in Internet-of-Things (IoT) networks has attracted increasing attention. By taking into account age-of-information (AoI), we investigate the trajectory planning problem of an unmanned aerial vehicle (UAV) that is used to aid a cluster-based IoT network. An optimization problem is formulated to minimize the total AoI of the collected data by the UAV from the ground IoT network. Since the total AoI of the IoT network depends on the flight time of the UAV and the data collection time at hovering points, we jointly optimize the selection of hovering points and the visiting order to these points. We exploit the state-of-the-art transformer and the weighted A*, which is a path search algorithm, to design a machine learning algorithm to solve the formulated problem. The whole UAV-IoT system is fed into the encoder network of the proposed algorithm, and the algorithm's decoder network outputs the visiting order to ground clusters. Then, the weighted A* is used to find the hovering point for each cluster in the ground IoT network. Simulation results show that the trained model by the proposed algorithm has a good generalization ability to generate solutions for IoT networks with different numbers of ground clusters, without the need to retrain the model. Furthermore, results show that our proposed algorithm can find better UAV trajectories with the minimum total AoI when compared to other algorithms.

Index Terms—AoI, IoT, transformer, trajectory optimization, UAV.

I. INTRODUCTION

THE use of unmanned aerial vehicles (UAVs) has attracted a lot of attention from academia and industry. Given their high maneuvering capability and mobility, UAVs can be used as wireless relays or mobile base stations to provide reliable communications and better coverage for ground devices [1].

Thanks to these advantages, UAVs can be flexibly deployed to provide fast and reliable network access in different applications, such as disasters [2], surveillance [3], monitoring [4], to name a few.

Since UAVs can fly close to the ground devices and provide low-altitude air-to-ground communication links with them, UAVs can be deployed to hover the area of interest to collect data from ground Internet-of-Things (IoT) networks. By doing so, UAV-aided data collection can save the energy of devices in traditional IoT networks, thus extending their lifetime [5]. However, maintaining the freshness of the collected information is an important issue in time-sensitive IoT applications, such as environmental monitoring and safety protection. In these applications, the generated data needs to be sent to the destination as soon as possible. Outdated information can lead to incorrect control and even cause major disasters [6]. Therefore, it is essential to ensure the freshness of the data received at the destination. To measure the freshness of information, the age of information (AoI) as a new performance metric was proposed in [7]. In a nutshell, AoI describes the amount of time elapsed since the generation of the most recent data update. AoI-based data collection can guarantee information freshness in IoT networks, which is quite different from traditional delay-based and throughput-based metrics [8]. As such, it has attracted increasing attention.

Due to the importance of AoI, a number of studies have been carried out on AoI-oriented data collection in UAV-assisted wireless networks. In [9], the authors aimed to minimize the average AoI of the system by optimizing the trajectory of the UAV in a UAV-aided data collection system. In [10], the authors optimized the trajectory of the UAV to minimize the maximal AoI and the average AoI of sensors. In [11], the authors assumed the UAV supports three modes to collect data and jointly optimize the trajectory and data collection modes of the UAV to minimize the average AoI of all ground nodes. In [12], the UAV trajectory, energy, and service time allocation were jointly optimized by an iterative algorithm in order to minimize the overall peak AoI of the system. The authors in [13] developed an energy-efficient navigation policy for the UAV to improve data freshness of the IoT network. In order to minimize the weighted sum of AoI, the authors in [14] jointly optimized the flight trajectory of the UAV and the transmission scheduling of sensors. From the above discussion,

it can be seen that AoI-oriented data collection problems in the UAV-assisted IoT network are typically related to UAV's trajectory design.

When collecting data in the UAV-assisted IoT network, if the UAV is dispatched to visit every ground IoT device, the energy consumption of the UAV will increase because of the increased UAV trajectory. Hence, to reduce the energy consumption of the UAV, clusters-based model have been extensively investigated in UAV-assisted wireless networks. For instance, in [15], to gather compressive data measurements, the authors divide the sensor network into multiple clusters. In each cluster, all nodes build a forwarding tree based on compressive data gathering to send data to the cluster head (CH). The UAV then traverses all CHs to collect the aggregated data. The authors jointly optimized the UAV trajectory, CH selection, and forward tree construction to minimize the total transmit power in the network. In [16], the authors consider a pre-clustered network where a UAV equipped with multiple antennas communicates with multiple ground users simultaneously, in a given time slot, using space division multiple access. The authors jointly optimized the time slot allocation and the UAV hovering time to minimize the overall energy consumption. In [17], the authors considered a UAV-enabled data collection system for massive machine-type communications (mMTC) where machine-type communication devices (MTCDs) are divided into several clusters. A UAV visits each hovering position which corresponds to a MTCD cluster and sequentially collects data from each MTCD in the corresponding cluster. They formulated a problem of minimizing the total energy consumption of the system. In our previous work [5], we considered using a UAV to collect data from a clustered IoT network, where the hovering points of the UAV are determined by the unknown CHs location. In other words, in [5], we jointly select the CHs and their visiting order to minimize the total energy consumption. In this paper, we examine the scenario where the UAV collects data from a group of clusters and the UAV only interact with the CHs. The problem of interest in this paper is to jointly optimize the UAV's hovering points and trajectory to achieve the minimal AoI data collection in a cluster-based IoT network. The optimization problem is formulated as a traveling salesman problem (TSP) with neighborhoods (TSPN), which is extremely challenging because it includes a continuous problem (optimization of hovering points) and a combinatorial problem (optimization of visiting order).

The hovering points of the UAV and the visiting order to these hovering points have a great impact on the flying time of the UAV and data collection time, which directly influence the total AoI of collected data. There have been some works on solving the TSPN efficiently. In [18], the Dubins TSPN was converted to a generalized TSP (GTSP) by using the sampling-based roadmap method, and then to an asymmetric TSP that can be addressed by the Lin-Kernighan heuristic algorithm. To handle the continuous optimization problem of waypoints within each circular neighborhood, the authors in [19] proposed a discretization scheme that equidistantly samples possible locations along the circular border of the interest neighborhood to determine the locations of the

waypoints. In this paper, in order to reduce the computational complexity for solving the joint optimization of the UAV's hovering points and trajectory to achieve the minimal AoI data collection in a cluster-based IoT network, we transform the formulated continuous optimization TSPN into a GTSP by borrowing the sampling-based idea. The transformed GTSP is a combinatorial optimization problem that can be solved using traditional methods, such as exact algorithms, approximate algorithms, or heuristic algorithms. However, these traditional algorithms may not achieve a good balance between optimality and computational complexity. Thus, by considering optimality, computational complexity, and generality, we shall develop a machine learning-based algorithm to solve the transformed GTSP, i.e., the UAV's trajectory design problem.

Machine learning has been explored as a promising technique for solving UAV's trajectory planning problems in UAV-assisted IoT networks. To minimize the weighted sum-AoI in a UAV-assisted network, the authors in [20] applied deep reinforcement learning (DRL) to optimize the UAV's trajectory using a deep Q network (DQN) and an artificial neural network (ANN). In [21], the authors utilized Q-learning to optimize AoI-optimal UAV path by considering the deadline constraints of data in the UAV-aided sensing network. In [22], the authors jointly optimized the UAV's trajectory and scheduling of the status update packets to minimize the normalized weighted sum of AoI in a UAV-assisted wireless network. Specifically, they used ANN, DQN, and long short-term memory (LSTM) to develop a DRL algorithm for learning the UAV trajectory in large-scale networks. Different from these works, we employ the state-of-the-art transformer and the weighted A* search method to design a UAV trajectory planning algorithm for AoI-oriented data collection.

Transformer was originally proposed by Google as a sequence-to-sequence model to deal with machine translation problem [23]. It has achieved great success in many areas of artificial intelligence in the past four years, such as computer vision, audio processing, document summarization, and document generation. Some researchers also attempt to use transformer and its variants to tackle combinatorial problems, such as the TSP. In [24], the cities in the TSP were encoded by a transformer and decoded sequentially through a query consisting of the last three cities in the partial tour. The used transformer was trained by reinforcement learning. In [25], the authors also used the transformer architecture as the encoder network and the decoder network outputs the result sequentially based on the embeddings from the encoder and the outputs generated at previous steps. The encoder and decoder networks were trained using a reinforce algorithm with a deterministic greedy baseline. The authors in [26] proposed a transformer-based framework to automatically learn improved heuristics on two representative routing problems: the TSP and capacitated vehicle routing problem (CVRP). In [27], the authors used the standard transformer architecture to tackle TSP and achieve an improved performance over recent learned heuristics. Inspired by the success of employing transformer in solving various problems of route planning, we propose the transformer-weighted-A* (TWA*) algorithm in this paper for solving our formulated GTSP combinatorial optimization

problem. Although the Ptr-A* algorithm proposed in our previous work [5] achieves good performance in solving the GTSP, the TWA* algorithm has the following two important advantages over the Ptr-A* algorithm. First, TWA* does not relay on past hidden states like Ptr-A*, and thus, avoids losing past information. Second, TWA* has the ability of parallel computation which makes it faster than Ptr-A*.

The main contributions of this paper are summarized as follows:

1) We propose an AoI-oriented data collection model in a cluster-based IoT network and formulate a total AoI-minimal trajectory planning problem where the hovering points of the UAV and the visiting order to these points are jointly optimized.

2) We view the formulated problem as a "machine translation" problem where the "source language" is the whole UAV-IoT network and the "target language" is the UAV trajectory with the minimal total AoI. The state-of-the-art TWA* is employed to solve the formulated problem. The parameters of the proposed algorithm are trained by reinforcement learning that only needs the reward calculation.

3) The learned policy by the proposed algorithm generalizes well on different sizes of problem instances. In other words, the trained model by the proposed algorithm can automatically find a trajectory with the minimal total AoI for new problem instances, without retraining the model.

4) Extensive simulations are conducted to evaluate the performance of the proposed algorithm. Results show that the proposed algorithm achieves significant performance gain in maintaining data freshness while reducing computation time when compared with other baseline algorithms.

The rest of this paper is organized as follows. Section II introduces the system model and presents the formulated problem. Section III develops the proposed algorithm. Section IV provides simulation results. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a UAV-assisted IoT network that consists of one rotary-wing UAV, one ground base station (BS) located at b_0, and M clusters of ground sensor nodes. Specifically, each cluster $m, m = 1, \ldots, M$, has one CH, located as b_m, and N_m ordinary sensor nodes, located at as $B_m = \{b_m^{(1)}, \ldots, b_m^{(N_m)}\}$. The ground IoT network performs some sensing tasks in the surrounding area where the ordinary sensor nodes are responsible for sampling data and forwarding the collected data to their corresponding CHs. The UAV is dispatched from the start hovering point c_0 which is directly above b_0 to visit M mission hovering points $\{c_1, \ldots, c_m, \ldots, c_M\}$ by a pre-designed trajectory for data collection, and then flies back to c_0 after completing the data collection task. Each hovering point corresponds to one ground cluster and its position will be determined by the proposed algorithm. The three-dimensional (3D) Cartesian coordinates system is
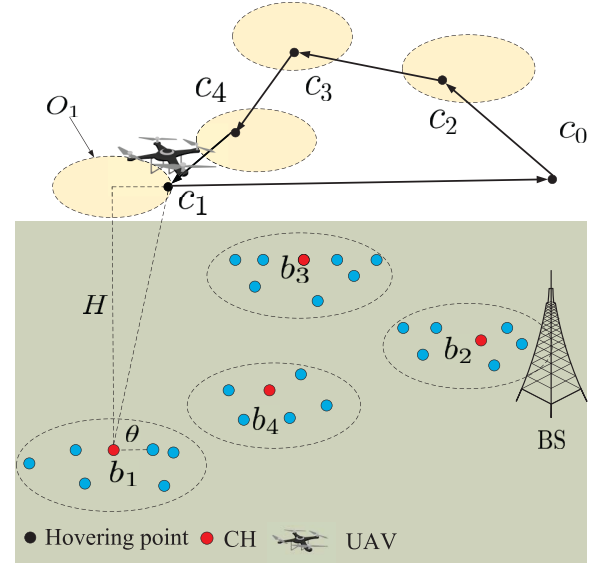

Fig. 1. System model of a UAV-assisted IoT network.

considered to define positions of hovering points and all CHs. The coordinate of the m-th hovering point is denoted by $c_m = (x_{c_m}, y_{c_m}, H) \in \mathbb{R}^3$, where H is the flight height of the UAV, whereas the location of the corresponding ground CH is given by $b_m = (x_{b_m}, y_{b_m}, 0) \in \mathbb{R}^3$.

We assume that the rotary-wing UAV supports a flying-hovering mode without considering acceleration-deceleration, i.e., it flies to the hovering points with a fixed speed v_{UAV} and hovers at these points with static status to collect data from ground CHs. We illustrate the UAV-assisted data collecting process in Fig. 1. The UAV takes off from c_0, determines the position of the hovering point c_2 that will be visited first and arrives at it. The UAV repeats this procedure until data collection of all clusters is completed, and flies back to c_0. Hence, the final trajectory of the UAV in this example is $\{c_0, c_2, c_3, c_4, c_1, c_0\}$.

A. Data Collection Model

When the UAV arrives at c_m, it sends a beacon message to wake up the corresponding CH b_m from its sleep mode. The beacon message includes the type of sensor nodes to be activated in response to the beacon, the data collection height of the UAV, a threshold to limit the number of sensor nodes in the CH (if necessary), and a trailer that has error detection capabilities. Then, b_m switches to its active mode and informs its member nodes in the same cluster to sample and send their sampled data sequentially according to the pre-allocated equal-length time slots using time-division multiplexing (TDM) protocol to avoid collision. We consider the *generate-at-will* model [28] as the data sampling model for all ordinary sensor nodes, by which nodes can generate information updates at any time. Specifically, we assume that each node can generate an update message of size L_{data} only in its allocated time slot to eliminate the waiting time. Also, each message has a time stamp, which is the start of each time slot. The length of a time slot is denoted as τ seconds. After the CH located at

b_m finishes collecting data from its member nodes, it will forward the collected data to the UAV. For ease of analysis, the wake-up time of nodes, including CHs and all ordinary nodes, and the information sampling time of each node are assumed negligible as compared to the data collection time. Thus, the data collection time of the UAV at each hovering point mainly consists of two parts: the data transmission time from ordinary nodes to their CHs and the time consumed for forwarding the collected data from CHs to the UAV.

We consider both the line-of-sight (LoS) and non-line-of-sight (NLoS) links to design the ground-to-air communication when the UAV hovers at mission hovering points. The LoS link probability is related to environment, elevation angle, and transmission distance, which can be expressed as [29]

$$P_{c_m}^{(\text{LoS})} = \frac{1}{1 + \beta \exp\left(-\widetilde{\beta}\left(\theta_{c_m} - \beta\right)\right)}, \tag{1}$$

where β and $\widetilde{\beta}$ are constants determined by the environment, $\theta_{c_m} = \arctan\left(H/R_{(c_m,b_m)}\right)$ is the elevation angle between b_m and the UAV when it hovers at c_m, $R_{(c_m,b_m)} = \sqrt{\left(x_{c_m} - x_{b_m}\right)^2 + \left(y_{c_m} - y_{b_m}\right)^2}$ is the horizontal distance between the CH b_m and the hovering point. Correspondingly, the probability of NLoS is given by $P_{c_m}^{(\text{NLoS})} = 1 - P_{c_m}^{(\text{LoS})}$. In addition, the path loss models of LoS and NLoS between the CH b_m and the UAV follow [30]

$$L_{c_m}^{(\text{LoS})} = 20 \log_{10}\left(\frac{4\pi f_c d_{(c_m,b_m)}}{v_{\text{light}}}\right) + \xi_{\text{LoS}}, \tag{2}$$

$$L_{c_m}^{(\text{NLoS})} = 20 \log_{10}\left(\frac{4\pi f_c d_{(c_m,b_m)}}{v_{\text{light}}}\right) + \xi_{\text{NLoS}}, \tag{3}$$

where f_c is the carrier frequency, v_{light} is the speed of light, $d_{(c_m,b_m)} = \sqrt{H^2 + R_{(c_m,b_m)}^2}$ is the distance between the UAV and the CH b_m, ξ_{LoS} and ξ_{NLoS} ($\xi_{\text{LoS}} < \xi_{\text{NLoS}}$) are the excessive path losses in LoS and NLoS links, respectively. We consider the average path loss to describe the link from the ground CH to the UAV, which can be expressed as

$$\overline{L}_{c_m} = P_{c_m}^{(\text{LoS})} L_{c_m}^{(\text{LoS})} + P_{c_m}^{(\text{NLoS})} L_{c_m}^{(\text{NLoS})}. \tag{4}$$

To avoid the interference among CHs, we assume that only one CH can transmit data to the UAV at any given time. Hence, the average available transmission rate in bits per second (bps) from CH b_m to the UAV can be expressed as $r_{c_m} = B_{\text{width}} \log_2\left(1 + \gamma_{c_m}\right)$, where B_{width} is the channel bandwidth in hertz (Hz), $\gamma_{c_m} = P_{\text{CH}}/\left(\sigma^2 10^{\overline{L}_{c_m}/10}\right)$ is the signal-to-noise ratio (SNR) of the transmission link, σ^2 is the noise power at the UAV, and P_{CH} is the transmission power of the CH. Regarding the transmission quality, we set a SNR threshold γ_{th} and the transmission is considered successful if the SNR is greater than the threshold. Thus, the SNR constraint at the UAV receiver is given as

$$\gamma_{c_m} \geq \gamma_{\text{th}}. \tag{5}$$

Lemma 1: Given the fixed flight height H, c_m should be located in a horizontal disk region centered at the position that directly above b_m and having the radius R^* which can

guarantee that the UAV successfully receives data. When $R_{(c_m,b_m)} = R^*$, the received SNR of the UAV at c_m is equal to γ_{th}.

Proof: See Appendix A.

Based on *Lemma 1*, we formally define a hovering disk region for each mission hovering point (excluding the start point c_0) as

$$O_m = \{c_m : ||c_m - b'_m|| = R_{(c_m,b_m)} \leq R^*\} \tag{6}$$

where $b'_m = (x_{b_m}, y_{b_m}, H) \in \mathbb{R}^3$ is the center of the disk O_m, and R^* is the radius to maintain a pre-defined quality-of-service, which can be found numerically. As long as the UAV enters a hovering disk region, it can collect data from the corresponding ground CH. The total data collection time of the UAV at $c_m \in O_m$ (or its hovering time) can be simply written as

$$T_{c_m}^{(\text{hov})} = N_m \tau + \frac{N_m L_{\text{data}}}{r_{c_m}} \tag{7}$$

where the first term in the right hand side is the time consumed for transmitting data from ordinary nodes to their corresponding CH b_m, and the second term is the data transmission time from b_m to the UAV. Therefore, the energy consumption of propulsion-related and communication-related activities of the UAV while hovering at c_m is expressed as

$$E_{c_m} = P_{\text{hov}} T_{c_m}^{(\text{hov})} + P_{\text{com}} \frac{N_m L_{\text{data}}}{r_{c_m}} \tag{8}$$

where P_{hov} and P_{com} are the UAV's powers for hovering and communication, respectively. After finishing the data collection task, b_m switches to the sleep model for saving energy. The UAV continues to select the next hovering point and executes the same processes to collect the sensed data from the corresponding ground cluster.

B. UAV's Mobility Model

Without loss of generality, the flight trajectory of the UAV can be seen as a permutation of the visiting order to M mission hovering points, with the start point being c_0, i.e., $\mathbf{c} = \{c_0, c_1, \ldots, c_M\}$. The set of all possible permutations is denoted as $\mathbf{\Phi}$ with the size of $M!$. We represent one of the permutations as $\boldsymbol{\pi} = \{\pi(0), \ldots, \pi(M+1)\}$ and express the ordered hovering points as $\mathbf{c}_{\boldsymbol{\pi}} = \{c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(M)}, c_{\pi(M+1)}\}$, where $c_{\pi(t)}, t = 0, \ldots, M+1$, is the hovering point that is visited at step t in the trajectory, and $c_{\pi(0)} = c_{\pi(M+1)} = c_0$. For ease of understanding, if the hovering point c_m is visited at step t, its corresponding cluster of ground ordinary nodes (B_m) and the number of ordinary nodes (N_m) are redefined as $B_{\pi(t)}$ and $N_{\pi(t)}$, respectively.

After finishing data collection at $c_{\pi(t)}$ with the hovering model, the UAV horizontally flies to the next hovering point $c_{\pi(t+1)}$ along the line segment connecting $c_{\pi(t)}$ and $c_{\pi(t+1)}$. The flying time of the UAV during this period is given by

$$T_{(c_{\pi(t)},c_{\pi(t+1)})}^{(\text{fly})} = \frac{||c_{\pi(t)} - c_{\pi(t+1)}||}{v_{\text{UAV}}} \tag{9}$$

where $||c_{\pi(t)} - c_{\pi(t+1)}||$ is the Euclidean distance between $c_{\pi(t)}$ and $c_{\pi(t+1)}$.

Following [31], the propulsion power consumption of the UAV for horizontal movement is the function of speed v_{UAV} and given by

$$
P_{\text{mov}}(v_{\text{UAV}}) = P_0 \left(1 + \frac{3v_{\text{UAV}}^2}{U_{\text{tip}}^2} \right) + P_1 \left(\left(1 + \frac{v_{\text{UAV}}^4}{4v_0^4} \right)^{1/2} \right.
$$
$$
\left. - \frac{v_{\text{UAV}}^2}{2v_0^2} \right)^{1/2} + \frac{1}{2} d_0 \rho s_0 \delta v_{\text{UAV}}^3 \quad (10)
$$

where P_0 and P_1 represent, respectively, the blade profile power and induced power in the hovering state, U_{tip} is the tip speed of the rotor blade of the UAV, v_0 is the mean rotor induced velocity in the hovering state, d_0 denotes the fuselage drag ratio, s_0 represents the rotor solidity, ρ is the density of air, and δ denotes the area of the rotor disk. According to the analysis in [31], the power consumption $P_{\text{mov}}(v_{\text{UAV}})$ firstly decreases and then increases with the increasing value of the speed v_{UAV}. The energy consumption in the UAV's flight from $c_{\pi(t)}$ to $c_{\pi(t+1)}$ is computed as

$$
E_{(c_{\pi(t)}, c_{\pi(t+1)})} = P_{\text{mov}}(v_{\text{UAV}}) T_{(c_{\pi(t)}, c_{\pi(t+1)})}^{(\text{fly})}. \quad (11)
$$

In the hovering state, the power consumption of the UAV can be obtained by substituting $v_{\text{UAV}} = 0$ into (10), $P_{\text{hov}} = P_0 + P_1$, which is a constant value.

C. Age of Information Model in a UAV-IoT System

We use the AoI metric to measure the freshness of information. According to the definition of AoI in [32], the AoI of a packet collected from node $b_{\pi(t)}^{(n)}$ in the $\pi(t)$-th visited cluster at time ζ is defined as

$$
A_{\pi(t)}^{(n)}(\zeta) = \left(\zeta - u_{\pi(t)}^{(n)}(\zeta) \right)^+ \quad (12)
$$

where $u_{\pi(t)}^{(n)}(\zeta)$ is the instant at which the packet is generated, and $(x)^+ = \max\{0, x\}$. When $\zeta < u_{\pi(t)}^{(n)}(\zeta)$, we define $A_{\pi(t)}^{(n)}(\zeta) = 0$. This is because the packet of node $b_{\pi(t)}^{(n)}$ has not been sampled. It is evident that the AoI of a packet will increase with time. In the considered UAV-IoT system, the BS is seen as the observer, thus, the AoI of a data packet can be seen as the amount of time elapsed from the instant at which the packet is generated to the instant at which the UAV flies back with the collected data to the BS.

For ease of analysis, for any ordinary node $b_{\pi(t)}^{(n)}, n = 1, \ldots, N_{\pi(t)}$ in the $\pi(t)$-th visited cluster, the AoI of its packet can be simply divided into two components. The first component is the time needed for the CH of its associated cluster to collect data from $b_{\pi(t)}^{(n)}$ and other nodes whose data have not been gathered (i.e., nodes $b_{\pi(t)}^{(n+1)}, \ldots, b_{\pi(t)}^{(N_{\pi(t)})}$) and forward the collected data to the UAV. The second component is the time consumed by the UAV to carry the packet of $b_{\pi(t)}^{(n)}$ to the end point $c_{\pi(M+1)}$. Specifically, this period includes the flight time of the UAV to unvisited ground clusters and the data collection time in these clusters. For example, after completing the data collection at $c_{\pi(t)}$, the UAV will fly to the next hovering point $c_{\pi(t+1)}$ and gather information from the corresponding cluster. During this period, the AoI of the packet
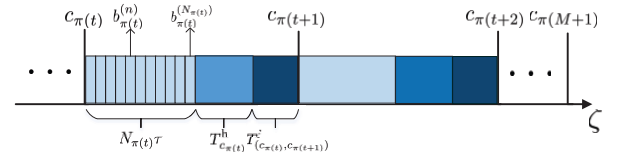

Fig. 2. The time sequence of data collection in the considered UAV-IoT system.

of $b_{\pi(t)}^{(n)}$ increases with time, which is the sum of the flight time $T_{(c_{\pi(t)}, c_{\pi(t+1)})}$ from $c_{\pi(t)}$ to $c_{\pi(t+1)}$ and data collection time $T_{c_{\pi(t+1)}}$ at hovering point $c_{\pi(t+1)}$. Then, the UAV performs the same process to unvisited clusters until it returns to the end point. The time sequence of data collection in the UAV-IoT system is illustrated in Fig. 2. Mathematically, the total AoI of the packet generated by $b_{\pi(t)}^{(n)}$ in the UAV-IoT system is given as

$$
A_{\pi(t)}^{(n)} = \underbrace{\left(N_{\pi(t)} - (n-1) \right) \tau + \frac{N_{\pi(t)} L_{\text{data}}}{r_{c_{\pi(t)}}}}_{\text{first component}}
$$
$$
+ \underbrace{\sum_{g=t}^{M-1} \left(T_{(c_{\pi(g)}, c_{\pi(g+1)})}^{(\text{fly})} + T_{c_{\pi(g+1)}}^{(\text{hov})} \right) + T_{(c_{\pi(M)}, c_{\pi(M+1)})}^{(\text{fly})}}_{\text{second component}}
$$
$$
\quad (13)
$$

which can be further simplified as

$$
A_{\pi(t)}^{(n)} = \sum_{g=t}^{M} \left(T_{c_{\pi(g)}}^{(\text{hov})} + T_{(c_{\pi(g)}, c_{\pi(g+1)})}^{(\text{fly})} \right) - (n-1)\tau. \quad (14)
$$

For packets of nodes in the same cluster, we have

$$
A_{\pi(t)}^{(1)} > A_{\pi(t)}^{(2)} \cdots > A_{\pi(t)}^{(N_{\pi(t)})}. \quad (15)
$$

On the other hand, the AoIs of packets in different clusters should satisfy

$$
A_{\pi(1)}^{(n)} > A_{\pi(2)}^{(n)} > \cdots > A_{\pi(M)}^{(n)}. \quad (16)
$$

D. Problem Formulation

The total AoI of all ordinary nodes in the network can be computed as

$$
\overline{A} = \sum_{t=1}^{M} \sum_{n=1}^{N_{\pi(t)}} A_{\pi(t)}^{(n)}
$$
$$
= \sum_{t=1}^{M} \sum_{n=1}^{N_{\pi(t)}} \sum_{g=t}^{M} \left(T_{c_{\pi(g)}}^{(\text{hov})} + T_{(c_{\pi(g)}, c_{\pi(g+1)})}^{(\text{fly})} \right)
$$
$$
- \sum_{t=1}^{M} \sum_{n=1}^{N_{\pi(t)}} (n-1)\tau. \quad (17)
$$

According to (17), the total AoI is expressed as a weighted sum of the flight time of the UAV and the data collection time at each hovering point, which is determined by the locations of hovering points c, the visiting order to these hovering points π. It is evident that the hovering points of the UAV and its trajectory have a strong impact on the total AoI of

data. If the position of any hovering point c_m is close to the center of the disk region O_m, a high data transmission rate can be achieved. As a result, the data transmission time from CHs to the UAV can be reduced, even though the UAV may have a longer flight trajectory, and hence the flight time. Conversely, if the UAV is located near the boundary of the disk region, the length of the UAV's trajectory might be reduced, but it will result in a lower data transmission rate, and hence increased data transmission time.

Our objective is to jointly find the hovering point from each disk and plan the visiting order to these hovering points for the UAV to minimize the total AoI of data in the considered UAV-IoT system. The optimization problem is expressed as follows:

$$\mathcal{P}_1 : \min_{c, \pi} \quad \overline{A}(c, \pi), \tag{18a}$$
$$\text{s.t.} \quad \pi \in \Phi, \tag{18b}$$
$$(5), (7), (9), (15), \text{ and } (16).$$

Constraint (18b) is the trajectory constraint. The SNR constraint is given in (5), and (7) is the data collection constraint. The flight time constraint is expressed as (9). AoI constraints are (15) and (16). It is evident that the formulated problem $\mathcal{P}_1$ is a TSPN [33], which combines the determination of hovering points at each disk with the problem of trajectory planning of the UAV. The traditional TSPN problem involves finding a minimum-cost tour (i.e., the total length of the tour is minimum) that travels each region exactly once for a collection of compact regions before returning to the initial departure point [34]. However, our formulated problem not only considers the traveling cost but also the cost spent at each hovering point. The problem $\mathcal{P}_1$ is extremely challenging because it is composed of a continuous problem (optimization of hovering points c) and a combinatorial problem (optimization of visiting order π). Given a set of hovering points c, the optimization of π can be viewed as the TSP, which can be normally be solved quite effectively by some dedicated TSP solvers, such as Concorde [35], etc. However, the optimization of hovering points c consists of an infinite number of variables, which is infeasible to be solved optimally. To reduce computational time, we leverage the sampling approach that samples finite discrete sets of hovering points from a continuous state space to transform the continuous TSPN in $\mathcal{P}_1$ into the GTSP. Specifically, each disk O_m is equally partitioned into $L_{\text{sub}} \times L_{\text{sub}}$ sub-regions and the center of each sub-region is selected as the possible hovering point. For some marginal sub-regions with non-square shape, we choose the centers of their actual areas. Hence, we can obtain a cluster G_m of sampling points with the size of L_{sub}^2 from O_m. As a result, our objective is changed to jointly select hovering points from M clusters of sampled hovering points and plan the UAV's trajectory to visit selected hovering points exactly once to minimize the total AoI. Using the sampling approach, the formulated problem $\mathcal{P}_1$ is converted to

$$\mathcal{P}_2 : \min_{c, \pi} \quad \overline{A}(c, \pi), \tag{19a}$$
$$\text{s.t.} \quad c_m \in G_m, G_m \in O_m, m \in \{1, \ldots, M\}, \tag{19b}$$
$$(5), (7), (9), (15), (16), \text{ and } (18b).$$

Obviously, the formulated problem $\mathcal{P}_2$ is a combinatorial optimization problem, and hence, NP-hard. There are two traditional methods to handle combinatorial problems: exact algorithms and heuristic algorithms. Exact algorithms can find optimal solutions, but they will become intractable when the size of problems grows. Heuristic algorithms' complexity is polynomial and they commonly find sub-optimal solutions. In contrast, we cast the proposed GTSP as a sequence-to-sequence problem where the source sequence is a set of clusters of hovering points and CHs and the target sequence is a set of selected hovering points and the visiting order to these points. We adopt the transformer, the weighted A*, and reinforcement learning to efficiently solve this problem.

III. TRANSFORMER-WEIGHTED A* ALGORITHM

Because the UAV needs to sequentially collect data from each ground cluster in the IoT network, we view the problem of the total AoI-minimal trajectory planning as a "machine translation" problem that is common in natural language processing. The whole UAV-IoT network as the "source language" is translated into the "target language", i.e., the UAV trajectory, by using our proposed TWA* algorithm. The TWA* algorithm is composed of an encoder network, a decoder network, and the weighted A* search algorithm which can effectively find the trajectory policy from hidden patterns behind a large number of training datasets.

A. Encoder

The role of the encoder network is to take the UAV-IoT network represented as an input sequence and map it into an abstract representation that is the learned information. The input sequence includes the start point of the UAV, each and every CH, number of nodes in each ground cluster, and all sampling points from each hovering disk. Specifically, we define $\boldsymbol{h}_0^{(\text{in})} = c_0 \in \mathbb{R}^3$, and $\boldsymbol{h}_m^{(\text{in})} = (G_m, b_m, N_m) \in \mathbb{R}^{3(L_{\text{sub}}^2+1)+1}, m \in \{1, \ldots, M\}$, where the cluster G_m of sampling points is represented as a $3L_{\text{sub}}^2$-dimensional vector as it includes L_{sub}^2 points with 3D Cartesian coordinates, the CH b_m is a 3-dimensional vector, and the number of nodes N_m is a constant. Hence, the input can be expressed as $\boldsymbol{H}^{(\text{in})} = \left(\boldsymbol{h}_0^{(\text{in})}; \boldsymbol{h}_1^{(\text{in})}; \ldots; \boldsymbol{h}_M^{(\text{in})}\right)$. The encoder network used in this paper is the standard transformer encoder with one embedding layer and six identical encoder layers as in [27]. Each encoder layer is composed of one multi-head self attention sub-layer and one point-wise feed-forward network sub-layer. Each sub-layer adds a residual connection and layer normalization. The embedding layer is to map each element of input to the d_{em}-dimensional vector space by a learnable linear projection. Specifically, to enable the model to distinguish the start point of the UAV from clusters, we separately utilize different parameters to compute the embeddings of the start point and the other clusters as follows:

$$\boldsymbol{h}_m^{(0)} = \begin{cases} \boldsymbol{W}_0 \boldsymbol{h}_m^{(\text{in})} + \boldsymbol{W}_{b_0}, & m = 0 \\ \boldsymbol{W}_1 \boldsymbol{h}_m^{(\text{in})} + \boldsymbol{W}_b, & m = 1, \ldots, M \end{cases} \tag{20}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{em}} \times 3}$, $\boldsymbol{W}_1 \in \mathbb{R}^{d_{\text{em}} \times (3(L_{\text{sub}}^2+1)+1)}$, $\boldsymbol{W}_{b_0} \in \mathbb{R}^{d_{\text{em}}}$, and $\boldsymbol{W}_b \in \mathbb{R}^{d_{\text{em}}}$ are learnable parameters. Then, the

embeddings $\boldsymbol{H}^{(0)} = \left(\boldsymbol{h}_0^{(0)}; \boldsymbol{h}_1^{(0)}; \ldots; \boldsymbol{h}_M^{(0)}\right) \in \mathbb{R}^{(M+1)\times d_{\text{em}}}$ are fed into the encoder layers. Note that we do not consider the positional decoding used in the original transformer in [23] because the order of the input sequence is irrelevant to the GTSP.

The attention layer in each encoder layer uses the multi-head self-attention mechanism with 8 heads to jointly attend to information from different representation subspaces at different positions. The 8 heads perform the attention calculation in parallel and their results are merged to produce an input for the next step. In the encoder layer $l, l = 1, \ldots, 6$, the output of self-attention on the h-th head, $h = 1, \ldots, 8$, is computed as

$$
\begin{aligned}
\boldsymbol{Z}_h^{(l)} &= \text{Attention}(\boldsymbol{Q}_h^{(l)}, \boldsymbol{K}_h^{(l)}, \boldsymbol{V}_h^{(l)}) \\
&= \text{softmax}\left(\frac{\boldsymbol{Q}_h^{(l)} \boldsymbol{K}_h^{(l)^T}}{\sqrt{d_\text{v}}}\right) \boldsymbol{V}_h^{(l)}
\end{aligned} \tag{21}
$$

where d_v is used for scaling the dot products, $\boldsymbol{Q}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$, $\boldsymbol{K}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$, and $\boldsymbol{V}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$ are matrices query, key, and value for the h-th head, respectively. They can be created by projecting the input query $\boldsymbol{Q}^{(l)}$, key $\boldsymbol{K}^{(l)}$, and value $\boldsymbol{V}^{(l)}$ of multi-head self attention with three learnable weight matrices $\boldsymbol{W}_h^{Q(l)} \in \mathbb{R}^{d_{\text{em}}\times d_\text{v}}$, $\boldsymbol{W}_h^{K(l)} \in \mathbb{R}^{d_{\text{em}}\times d_\text{v}}$, and $\boldsymbol{W}_h^{V(l)} \in \mathbb{R}^{d_{\text{em}}\times d_\text{v}}$, respectively, as follows

$$
\boldsymbol{Q}_h^{(l)} = \boldsymbol{Q}^{(l)} \boldsymbol{W}_h^{Q(l)}, \boldsymbol{K}_h^{(l)} = \boldsymbol{K}^{(l)} \boldsymbol{W}_h^{K(l)}, \boldsymbol{V}_h^{(l)} = \boldsymbol{V}^{(l)} \boldsymbol{W}_h^{V(l)} \tag{22}
$$

where $\boldsymbol{Q}^{(l)} = \boldsymbol{K}^{(l)} = \boldsymbol{V}^{(l)} = \boldsymbol{H}^{(l-1)}$. In this paper $\boldsymbol{H}^{(l-1)}$ is the output of the encoder layer $(l-1)$ or the output of the embedding layer before the encoder layer 1. Matrices $\boldsymbol{Q}_h^{(l)}$, $\boldsymbol{K}_h^{(l)}$, and $\boldsymbol{V}_h^{(l)}$ can be further expressed as

$$
\boldsymbol{Q}_h^{(l)} = \begin{pmatrix} \boldsymbol{q}_0 \\ \vdots \\ \boldsymbol{q}_M \end{pmatrix}, \boldsymbol{K}_h^{(l)} = \begin{pmatrix} \boldsymbol{k}_0 \\ \vdots \\ \boldsymbol{k}_M \end{pmatrix}, \boldsymbol{V}_h^{(l)} = \begin{pmatrix} \boldsymbol{v}_0 \\ \vdots \\ \boldsymbol{v}_M \end{pmatrix} \tag{23}
$$

where $\forall \boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} \in \mathbb{R}^{d_\text{v}}$. Then, we can obtain the scaled attention scores

$$
\begin{aligned}
\frac{\boldsymbol{Q}_h^{(l)} \left(\boldsymbol{K}_h^{(l)}\right)^T}{\sqrt{d_\text{v}}} &= \frac{1}{\sqrt{d_\text{v}}} \begin{pmatrix} (\boldsymbol{q}_0, \boldsymbol{k}_0) & \cdots & (\boldsymbol{q}_0, \boldsymbol{k}_M) \\ (\boldsymbol{q}_1, \boldsymbol{k}_0) & \cdots & (\boldsymbol{q}_1, \boldsymbol{k}_M) \\ \cdots & (\boldsymbol{q}_i, \boldsymbol{k}_j) & \cdots \\ (\boldsymbol{q}_M, \boldsymbol{k}_0) & \cdots & (\boldsymbol{q}_M, \boldsymbol{k}_M) \end{pmatrix} \\
&= \begin{pmatrix} u_{00} & \cdots & u_{0M} \\ u_{10} & \cdots & u_{1M} \\ \cdots & u_{ij} & \cdots \\ u_{M0} & \cdots & u_{MM} \end{pmatrix}
\end{aligned} \tag{24}
$$

where $(\boldsymbol{q}_i, \boldsymbol{k}_j), i, j \in \{0, \ldots, M\}$ is the inner product of vectors, which measures the similarity of vector $\boldsymbol{q}_i$ and vector $\boldsymbol{k}_j$. The row-wise *softmax* function is used on each element of the above scaled attention scores matrix, which is given by $\overline{u}_{ij} = e^{u_{ij}} / \sum_{j'=0}^M e^{u_{ij'}}$. Then, the output $\boldsymbol{Z}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$

of the h-th head is expressed as

$$
\boldsymbol{Z}_h^{(l)} = \begin{pmatrix} \sum_{j=0}^M \overline{u}_{0j} \boldsymbol{v}_j \\ \sum_{j=0}^M \overline{u}_{1j} \boldsymbol{v}_j \\ \vdots \\ \sum_{j=0}^M \overline{u}_{Mj} \boldsymbol{v}_j \end{pmatrix} = \begin{pmatrix} \overline{\overline{\boldsymbol{u}}}_0 \\ \overline{\overline{\boldsymbol{u}}}_1 \\ \vdots \\ \overline{\overline{\boldsymbol{u}}}_M \end{pmatrix}. \tag{25}
$$

Hence, we end up with 8 different outputs from 8 heads where each head could learn something different. These outputs are concatenated and multiplied by an additional learnable weight matrix $\boldsymbol{W}_\text{o}^{(l)} \in \mathbb{R}^{8d_\text{v}\times d_{\text{em}}}$ to generate the final output of the multi-head attention layer, as follows:

$$
\boldsymbol{Z}^{(l)} = \left(\boldsymbol{Z}_1^{(l)}, \ldots, \boldsymbol{Z}_8^{(l)}\right) \boldsymbol{W}_\text{o}^{(l)}, \boldsymbol{Z}^{(l)} \in \mathbb{R}^{(M+1)\times d_{\text{em}}}. \tag{26}
$$

To facilitate the understanding of multi-head attention layer, all operations from (21) to (26) are defined as a function $\text{MHA}(\cdot)$. Thus, $\boldsymbol{Z}^{(l)} = \text{MHA}\left(\boldsymbol{Q}^{(l)}, \boldsymbol{K}^{(l)}, \boldsymbol{V}^{(l)}\right)$. Then, $\boldsymbol{Z}^{(l)}$ is added to the input of the multi-head attention in this encoder, which is a residual connection operation. Subsequently, the output of the residual connection is fed into a batch normalization, defined as a function $\text{BN}(\cdot)$, and it is written as $\boldsymbol{Z}'^{(l)} = \text{BN}\left(\boldsymbol{H}^{(l-1)} + \boldsymbol{Z}^{(l)}\right)$. The use of the residual connection is to avoid the degradation problem of the network in training, while the layer normalization can improve the training speed and the stability of the networks. The normalized residual output goes through a pointwise feed-forward network (defined as a function $\text{FFN}(\cdot)$), which is a couple of linear layers with a ReLU activation in between. Then, the output of the pointwise feed-forward network is added to its input by a residual connection and further normalized to obtain the final output $\boldsymbol{H}^{(l)} \in \mathbb{R}^{(M+1)\times d_{\text{em}}}$ of the encoder layer l, which is given by $\boldsymbol{H}^{(l)} = \text{BN}\left(\boldsymbol{Z}'^{(l)} + \text{FFN}\left(\boldsymbol{Z}'^{(l)}\right)\right)$. In each encoder layer, we perform the same computational process and finally output the final result of the encoder part at layer 6, $\boldsymbol{H}^{(6)} = \left(\boldsymbol{h}_0^{(6)}; \boldsymbol{h}_1^{(6)}; \ldots; \boldsymbol{h}_M^{(6)}\right) \in \mathbb{R}^{(M+1)\times d_{\text{em}}}$, which is the continuous representation with attention information of the input $\boldsymbol{H}^{(\text{in})}$. All of these operations will help the decoder network focus on the appropriate elements in the input during the decoding process.

B. Decoder

The decoding is autoregressive and generates the result one by one. The output of the decoder network can be represented as an ordered sequence of the input of the encoder. The decoder begins with the start point at decoding step 0 since the trajectory of the UAV should start at the start point as well as end at this point. The output of each decoding step is based on the information from the encoder and the already-generated previous output in the decoder. Hence, the decoding process can be modelled using the probability chain rule:

$$
P(\boldsymbol{\pi}|\boldsymbol{H}^{(\text{in})}) = \prod_{t=0}^{M+1} P(\pi(t)|\pi(0), \ldots, \pi(t-1), \boldsymbol{H}^{(\text{in})}). \tag{27}
$$

The decoding process aims at finding the optimal $\boldsymbol{\pi}$ to maximize $P(\boldsymbol{\pi}|\boldsymbol{H}^{(\text{in})})$.

The decoder network is composed of two identical decoder layers, and a single-head attention layer. Each decoder layer contains two multi-head attention sub-layers which employ a residual connection around them followed by layer normalization. These sub-layers have the same structure as the sub-layers in the encoder network but each of them has a different job. Since the output of the decoder network is related to the order, we need to inject some information about the positions into the input sequence of the decoder network. The locations are implicitly represented by the order of the data input to the decoder network. Hence, the input of the decoder network is the output of the encoder network combined with the positional encoding. Suppose the outputs of the decoder network at previous t decoding steps are $\pi(0), \pi(1), \ldots, \pi(t)$, the decoder wants to predict the output at $t+1$ step. Then, the input to the decoder network is expressed as $\widehat{\boldsymbol{H}}_{t+1}^{(0)} = \left(\widehat{\boldsymbol{h}}_{\pi(0)}^{(0)}; \widehat{\boldsymbol{h}}_{\pi(1)}^{(0)}; \ldots; \widehat{\boldsymbol{h}}_{\pi(t)}^{(0)} \right)$. Each element in $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ can be calculated by $\widehat{\boldsymbol{h}}_{\pi(t)}^{(0)} = \boldsymbol{h}_{\pi(t)}^{(6)} + \text{PE}_t$, where $\boldsymbol{h}_{\pi(t)}^{(6)} \in \mathbb{R}^{1 \times d_{\text{em}}}$ is one element in $\boldsymbol{H}^{(6)}$ which is decoded at the t-th step, $\text{PE}_t \in \mathbb{R}^{1 \times d_{\text{em}}}$ is the positional encoding based on the sinusoidal function, which is given by [36]

$$\text{PE}_t(d_i) = \begin{cases} \sin\left(\omega_{d_i} t\right), & \text{if } d_i \text{ is even} \\ \cos\left(\omega_{d_i} t\right), & \text{if } d_i \text{ is odd} \end{cases} \quad (28)$$

where d_i is the dimension, $1 \leqslant d_i \leqslant d_{\text{em}}$, ω_{d_i} is the hand-crafted frequency for each dimension. The position encoding of each position successfully provides the position information to the decoder network. The input $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ gets fed into the first multi-head attention sub-layer of the first decoder layer and pass through the residual connection and layer normalization (denoted as a function $\text{LN}(\cdot)$) to prepare the query for the next multi-head attention sub-layer, as follows

$$\widehat{\boldsymbol{Z}}_{t+1}^{(1)} = \text{MHA}\left(\widehat{\boldsymbol{h}}_{\pi(t)}^{(0)}, \widehat{\boldsymbol{H}}_{t+1}^{(0)}, \widehat{\boldsymbol{H}}_{t+1}^{(0)} \right), \widehat{\boldsymbol{Z}}_{t+1}^{(1)} \in \mathbb{R}^{1 \times d_{\text{em}}} \quad (29)$$

$$\widehat{\boldsymbol{Z}}_{t+1}^{'(1)} = \text{LN}\left(\widehat{\boldsymbol{h}}_{\pi(t)}^{(0)} + \widehat{\boldsymbol{Z}}_{t+1}^{(1)} \right), \widehat{\boldsymbol{Z}}_{t+1}^{'(1)} \in \mathbb{R}^{1 \times d_{\text{em}}} \quad (30)$$

where $\widehat{\boldsymbol{h}}_{\pi(t)}^{(0)}$ is the query, $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ works as the key and the value matrices in the current multi-head attention sub-layer. The second multi-head attention sub-layer is used to match the encoder's input to the decoder's input to allow the decoder network to decide the next possible output among the non-visited elements. For this sub-layer, the encoder network's output $\boldsymbol{H}^{(6)}$ is the key and the value matrices, and $\widehat{\boldsymbol{Z}}^{'(1)}$ is the query matrix. The calculations are given by

$$\widehat{\boldsymbol{Z}}_{t+1}^{''(1)} = \text{MHA}\left(\widehat{\boldsymbol{Z}}_{t+1}^{'(1)}, \boldsymbol{H}^{(6)}, \boldsymbol{H}^{(6)} \right), \widehat{\boldsymbol{Z}}_{t+1}^{''(1)} \in \mathbb{R}^{1 \times d_{\text{em}}} \quad (31)$$

$$\widehat{\boldsymbol{H}}_{t+1}^{(1)} = \text{LN}\left(\widehat{\boldsymbol{Z}}_{t+1}^{'(1)} + \widehat{\boldsymbol{Z}}_{t+1}^{''(1)} \right), \widehat{\boldsymbol{H}}_{t+1}^{(1)} \in \mathbb{R}^{1 \times d_{\text{em}}}. \quad (32)$$

Note that we add the mask of visited elements to the scaled attention scores in this sub-layer. Then, $\widehat{\boldsymbol{H}}_{t+1}^{(1)}$ goes through the second decoder layer to get the output $\widehat{\boldsymbol{H}}_{t+1}^{(2)} \in \mathbb{R}^{1 \times d_{\text{em}}}$. In order for the decoder network to compute output probabilities $P(\pi(t+1)|\pi(0), \ldots, \pi(t), \boldsymbol{H}^{(6)})$, $\widehat{\boldsymbol{H}}_{t+1}^{(2)}$ and the output $\boldsymbol{H}^{(6)}$ of the encoder network get fed into a single-head attention to get a distribution over the non-visited elements,

which is given by [27]

$$\boldsymbol{P}_{t+1} = \text{softmax}\left(\tanh\left(\frac{\widehat{\boldsymbol{Q}}_{t+1} \widehat{\boldsymbol{K}}_{t+1}^T}{\sqrt{d_{\text{em}}}} \odot \mathcal{M}_{t+1} \right) \right) \quad (33)$$

where $\widehat{\boldsymbol{Q}}_{t+1} = \widehat{\boldsymbol{H}}_{t+1}^{(2)} \widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{K}}_{t+1} = \boldsymbol{H}^{(6)} \widehat{\boldsymbol{W}}_2, \widehat{\boldsymbol{W}}_1 \in \mathbb{R}^{d_{\text{em}} \times d_{\text{em}}}$ and $\widehat{\boldsymbol{W}}_2 \in \mathbb{R}^{d_{\text{em}} \times d_{\text{em}}}$ are learnable weight matrices, $\mathcal{M}_{t+1}$ is the mask of the visited elements considered in this layer, $\odot$ is the Hadamard product, and $\boldsymbol{P}_{t+1} \in \mathbb{R}^{1 \times (M+1)}$ is the distribution over the non-visited elements, which is composed of probability scores. Then, the output that will be selected is sampled from the distribution with three decoding methods:

1) Greedy: At each decoding step, this method greedily selects the element with the largest probability $P(\pi(t+1)|\pi(0), \ldots, \pi(t), \boldsymbol{H}^{(6)})$.

2) Random Sampling: This method randomly samples W_{sampling} solutions, where each solution includes fully visiting order, and selects the solution with the highest probability as the final result.

3) Beam Search: This method chooses the top W_{beam} possible solutions that have the highest probability at each step, where W_{beam} is the beam width. Those W_{beam} solutions will move to the next time step, and the process repeats. Then, we can obtain a tree of solutions of each step and the $\boldsymbol{\pi}$ that has the highest overall probability is picked as the final result.

We assume that the index of the highest probability score in $\boldsymbol{P}_{t+1}$ is selected with the greedy decoding as the output $\pi(t+1)$ at step $t+1$. Thus $\pi(t+1)$ points to the element at the same position of the input sequence $\boldsymbol{H}^{(\text{in})}$ of the encoder network, which is represented as $\boldsymbol{h}_{\pi(t+1)}^{(\text{it})}$. Then, the decoder network takes the encoding information of $\boldsymbol{h}_{\pi(t+1)}^{(\text{it})}$ from $\boldsymbol{H}^{(6)}$, i.e., $\boldsymbol{h}_{\pi(t+1)}^{(6)}$, and adds it with its position encoding to the list of the decoder input to continue decoding for the next step. Finally, we can obtain a set of the visiting order, $\boldsymbol{\pi}$. As shown in the example in Fig. 3, $\boldsymbol{H}^{(\text{in})} = (c_0; (G_1, b_1, N_1); (G_2, b_2, N_2); (G_3, b_3, N_3); (G_4, b_4, N_4))$ is the input to the encoder network and the decoder network outputs the final visiting order $\boldsymbol{\pi} = \{\pi(0), \pi(1), \pi(2), \pi(3), \pi(4)\}$ to elements in $\boldsymbol{H}^{(\text{in})}$.

C. Selection of Hovering Points

Given the visiting order $\boldsymbol{\pi}$, we know the visiting order to all hovering points clusters and construct a graph containing all of them, as illustrated in Fig. 3. Each layer of the graph is composed of one hovering points cluster. Then, we will calculate the path with the minimal total AoI starting from the start point (marked as $\pi(0)$ in the visiting order), going through each cluster G_m, and ending at the clone of the start point (marked as $\pi(M+1)$ in the visiting order). To guarantee that at most one hovering point is selected from each cluster, we assume that all edges between possible hovering points of consecutive clusters to be directed by $\boldsymbol{\pi}$. We use the weighted A* search algorithm [37] to quickly find the hovering point from each cluster to build the path with the minimal cost (total AoI). We assume that the UAV currently reaches the point s' and will decide the next point to be expanded by the following
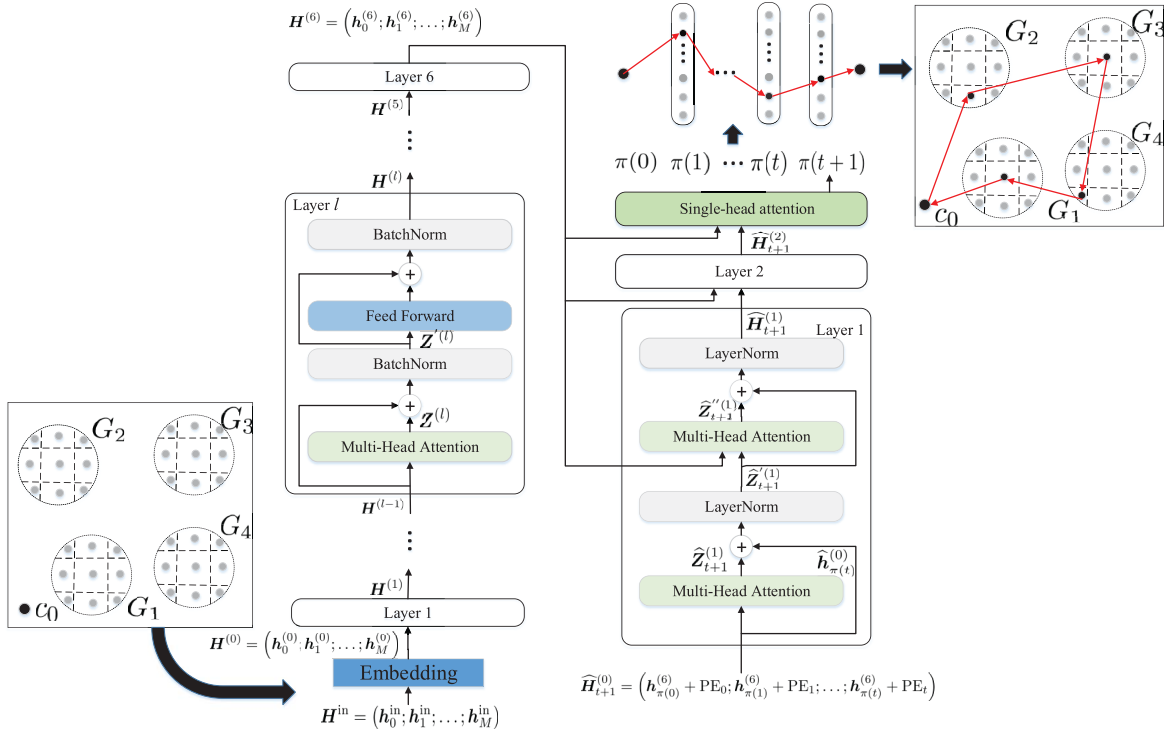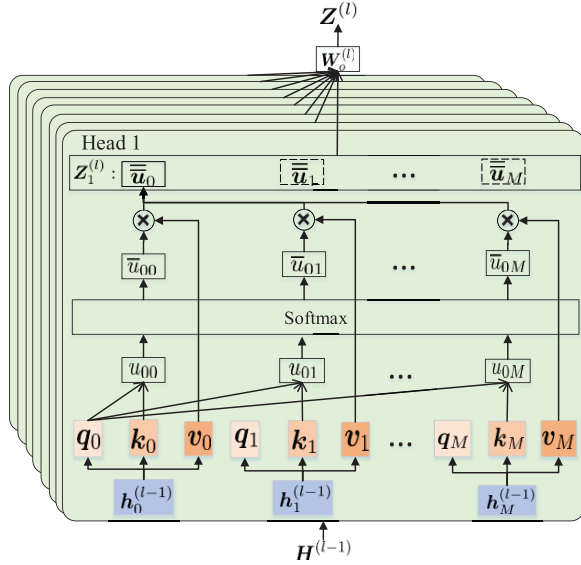
Fig. 3. The proposed algorithm framework.

Fig. 4. Multi-head self attention.

cost function

$$f(s) = g(s) + \omega h(s) \tag{34}$$

where s is any neighbor point of s', $g(s)$ is the total movement cost on the path from the start point c_0 to s, $h(s)$ is the heuristic function to estimate cost from s to the end point c_0', and $\omega > 1$ is a constant factor. The neighbor point with a minimal $f(s)$ value is expanded. The pseudocode is described in Algorithm 1. We use COST and FRONTIER to keep track of $g(s)$ and the expanding process, respectively. Each point that has been reached keeps a pointer to its parent in

CAME_FROM so that we can know where it came from. With CAME_FROM, we can construct a path having the minimal AoI from the start point to the end point, as illustrated by the solid red line with arrow in the example in Fig. 3.

D. Computational Complexity Analysis

In the encoder network, each encoder layer is the standard transformer encoder with quadratic computational complexity $O((M+1)^2 d_{\text{em}})$ [23]. Since the number of layers is constant, the computational complexity of the encoder network is still $O((M+1)^2 d_{\text{em}})$. In the decoder network, although each decoder layer contains two multi-head attention sub-layers, its computational complexity is still estimated to be quadratic $O((M+1)^2 d_{\text{em}})$ [23]. Likewise, the number of decoder layers does not affect the computational complexity of the decoder network. In addition, the computational complexity of the single-head attention used in the final step of the decoder network is also quadratic $O((M+1)^2 d_{\text{em}})$ [23]. Hence, the employed transformer model has the computational complexity $O((M+1)^2 d_{\text{em}})$, which is quadratic in the length of the input sequence. Different data structures used to implement the weighted A* algorithm, and hence, affect its computational complexity. We use the min heap to implement the weighted A* algorithm. We assume that at most ML_{sub}^2 points (the total number of points in the search graph) are visited, and the min heap uses $O(\log(ML_{\text{sub}}^2))$ computational complexity to extract a point each time [38]. The weighted A* algorithm's computational complexity is estimated to be $O(ML_{\text{sub}}^2 \log(ML_{\text{sub}}^2))$. Hence, the computational complexity of the proposed algorithm is $O((M+1)^2 d_{\text{em}}) + O(ML_{\text{sub}}^2 \log(ML_{\text{sub}}^2))$.

Algorithm 1: Pseudocode for Weighted A* Search Algorithm to Find Hovering Points

---

Input: created graph
1: FRONTIER = PriorityQueue()
2: FRONTIER.put(c_0, 0)
3: CAME_FROM = [ ]
4: COST = [ ]
5: CAME_FROM[c_0] = None
6: COST[c_0] = 0
7: **while** FRONTIER is not empty **do**
8:    current point s' = FRONTIER.get()
9:    **if** $s' = c_0'$ **then**
10:     break
11:    **end if**
12:    **for** each neighbor s of s' **do**
13:     $g(s) = \text{COST}[s'] +$ the total AoI from s' to s
14:     **if** s not in COST or $g(s) < \text{COST}[s]$ **then**
15:      COST[s] = g(s)
16:      $f(s) = g(s) + \omega h(s)$
17:      FRONTIER.put(s, $f(s)$)
18:      CAME_FROM[s] = s'
19:     **end if**
20:    **end for**
21: **end while**
22: calculate $\overline{A}$ according to CAME_FROM

Algorithm 2: Training TWA* by REINFORCE With Rollout Baseline

---

Input: Epochs E_{epochs}, training steps S, batch size B_{size}
1: Initialize parameters ϑ, $\vartheta^{(\text{BL})} \leftarrow \vartheta$
2: **for** epoch = 1 to E_{epochs} **do**
3:    **for** step = 1 to S **do**
4:     $\boldsymbol{H}_i^{(\text{in})} \leftarrow$ generate instances() $\forall i \in \{1, \ldots, B_{\text{size}}\}$
5:     $\boldsymbol{\pi}_i \leftarrow$ Sampling solution$P_\vartheta\left(\cdot | \boldsymbol{H}_i^{(\text{in})}\right)$
6:     $\boldsymbol{\pi}_i^{(\text{BL})} \leftarrow$ Greedy solution$P_{\vartheta^{(\text{BL})}}\left(\cdot | \boldsymbol{H}_i^{(\text{in})}\right)$
7:     $\overline{A}_i \leftarrow$ weighted A* $\left(\boldsymbol{\pi}_i\right)$
8:     $\overline{A}_i^{(\text{BL})} \leftarrow$ weighted A* $\left(\boldsymbol{\pi}_i^{(\text{BL})}\right)$
9:     $\nabla_\vartheta J \leftarrow \sum_{i=1}^{B_{\text{size}}} \left(\overline{A}_i - \overline{A}_i^{(\text{BL})}\right) \nabla_\vartheta \log P_\vartheta\left(\boldsymbol{\pi}_i | \boldsymbol{H}_i^{(\text{in})}\right)$
10:     $\vartheta \leftarrow$ Adam $\left(\vartheta, \nabla_\vartheta J\right)$
11:    **end for**
12:    **if** t-test$(P_\vartheta(\cdot), P_{\vartheta^{(\text{BL})}}(\cdot)) < 5\%$ **then**
13:     $\vartheta^{(\text{BL})} \leftarrow \vartheta$
14:    **end if**
15: **end for**

E. Training

To enable the transformer model to produce the optimal $\boldsymbol{\pi}$, we use the well-known policy gradient approaches to train it. The transformer model is parameterized by ϑ, which includes all trainable variables in the encoder and the decoder networks. We regard the UAV as an agent to learn a good policy $\boldsymbol{\pi}$ to maximize long-term rewards by iteratively interacting with the environment to optimize parameter ϑ. At each step, the agent in a given state chooses an action by its decision policy, which actually is the mapping from states to actions.

1) State: The state consists of the environment encoded by the encoder network and the visited clusters before the current step in the decoder, which are $\boldsymbol{H}^{(6)}$ and $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ in the transformer, respectively.

2) Action: At each step, the agent makes an action $\pi(t)$ based on its state, which can be seen as the processes of the right-hand side of (27). Thus, we view all operations in the decoder network as the action.

3) Reward: The negative of the total AoI $\overline{A}$ in (17) is used as the reward.

Our objective for training is given by

$$J\left(\vartheta | \boldsymbol{H}^{(\text{in})}\right) = \mathbb{E}_{\boldsymbol{\pi} \sim P_\vartheta\left(\cdot | \boldsymbol{H}^{(\text{in})}\right)} \left(\overline{A}\right). \tag{35}$$

The gradient of (35) is calculated using the REINFORCE algorithm [39] with the greedy rollout baseline $\overline{A}^{(\text{BL})}$ [25]

$$\nabla_\vartheta J\left(\vartheta | \boldsymbol{H}^{(\text{in})}\right)$$
$$= \mathbb{E}_{\boldsymbol{\pi} \sim P_\vartheta\left(\cdot | \boldsymbol{H}^{(\text{in})}\right)} \left[\left(\overline{A} - \overline{A}^{(\text{BL})}\right) \nabla_\vartheta \log P_\vartheta\left(\boldsymbol{\pi} | \boldsymbol{H}^{(\text{in})}\right)\right] \tag{36}$$

where $\overline{A}$ is the cost of a solution that is obtained from the current training transformer model by sampling decoding. We set the greedy policy as the baseline policy in our model, and hence, $\overline{A}^{(\text{BL})}$ is the cost of a solution of the deterministic greedy decoding, which is used to eliminate variance during training. By doing so, the transformer model is trained to improve over its (greedy) self. The training process is summarized in Algorithm 2. In each training step, new instances are generated first (line 4). Then, the transformer model uses sampling decoding and greedy decoding to produce $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_i^{(\text{BL})}$ (lines 5 and 6), respectively. The total AoIs are further obtained from the weighted A* (lines 7 and 8). The gradient in (36) is approximated with Monte Carlo sampling in a batch size B_{size} (line 9). The model parameter ϑ is updated using the Adam optimizer (line 10). We compare the current policy with the greedy baseline policy and update the parameter $\vartheta^{(\text{BL})}$ only if the improvement is significant according to a paired t-test (5%) [25].

As pointed out earlier, Ptr-A* proposed in our previous work [5] can also be used to solve the formulated GTSP in this work. Here, we give comparisons between TWA* and Ptr-A* in detail. First, they have different structures. In Ptr-A*, the encoder network consists of the LSTM networks, and the decoder network is composed of the LSTM networks and the attention mechanism. The LSTM networks have the form of a chain of repeating modules of a neural network. The key to the LSTM networks is the cell state, which is the hidden state. In theory, the hidden state can carry relevant information throughout the processing of the sequence. Since the LSTM networks process the elements of the input sequence one by one, the hidden state of each element of the input sequence is calculated by the current element and the previous hidden state. The final hidden state of the encoder network is fed into the LSTM networks of the decoder network. Then, the attention mechanism uses the hidden state of the decoder network to generate the output sequence. In TWA*, the encoder network

includes six identical encoder layers in which each encoder layer is mainly composed of one multi-head self-attention sub-layer and one point-wise feed-forward network. The decoder network of TWA* consists of two identical decoder layers and one single-head attention layer. The encoder network is used to map the input $\boldsymbol{H}^{(0)} = \left(\boldsymbol{h}_0^{(0)}; \boldsymbol{h}_1^{(0)}; \ldots; \boldsymbol{h}_M^{(0)}\right)$ to a sequence of continuous representations. The decoder network receives the output of the encoder together with the decoder output at the previous time step to generate the output sequence. Multi-head self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Multi-head self-attention helps TWA* to look at all elements in the input sequence for clues that can help lead to a better encoding. Unlike hidden states used in Ptr-A*, TWA* does not rely on past hidden states to capture dependencies with previous elements in the sequence. As a result, TWA* does not suffer from long dependency issues, which are very common in recurrent-based networks, such as RNNs and LSTMs, and hence does not lost past information.

Second, TWA* can process sequences in parallel, which is faster than Ptr-A*. In Ptr-A*, the elements of a sequence must be processed one by one and each element's hidden state is assumed to be dependent only on the previously hidden state. Hence, Ptr-A*'s recurrent structure makes it hard to use parallel computing to process sentences and this means that it is very slow in training and inference. All elements in a sequence are processed in TWA* as a whole rather than one by one. Because of the use of multi-head self-attention that is designed in parallel, TWA* has the ability of parallel computation.

Third, the training method used in this work is different from the training method in [5]. In [5], in order to obtain the optimal parameter of Ptr-A*, we use the actor-critic architecture to train Ptr-A* where a second critic network must be trained. In this work, we use the greedy rollout baseline where the TWA* model is trained to improve over its (greedy) self. This training method can avoid all the inherent training difficulties associated with the actor-critic architecture.

IV. NUMERICAL RESULTS

We conduct extensive experiments to investigate the performance of the proposed TWA* algorithm in solving the problem of trajectory planning to minimize the total AoI for the UAV-IoT network. The proposed model is implemented by Pytorch 1.7 and Python 3.8 and trained on a machine with 1 NVIDIA RTX 2080Ti GPU.

A. Test Settings

1) Decoding Strategies: As we mentioned in Section III-D, the random sampling decoding and the greedy decoding are employed for training the model. At inference, we evaluate performance of all three decoding methods on test instances and they are marked as TWA*–greedy, TWA*–sampling ($W_{\text{sampling}} = 5120$), and TWA*–beam search ($W_{\text{beam}} = 100$).

TABLE I
SIMULATION PARAMETERS

| Parameter	Value	Parameter	Value
H	100 m	β	12.08
$\bar{\beta}$	0.11	P_{CH}	0.1 W
γ_{th}	20 dB (default)	ξ_{LoS}	1 dB
ξ_{NLoS}	20 dB	σ^2	-110 dBm
v_{UAV}	15 m/s	f_c	2 GHz
L_{data}	5 Mb	B_{width}	1 MHz
P_{com}	0.1 W	L_{sub}	5
P_0	99.66 W	P_1	120.16 W
U_{tip}	120 m/s	v_0	0.002 m/s
d_0	0.48	ρ	1.225 kg/m^3
τ	0.1 s	δ	0.5
s_0	0.0001		

2) Comparison Algorithms: To evaluate the effectiveness of the proposed model with different decoding methods, we compare it with the genetic algorithm [40], the simulated annealing (SA) algorithm [41], and Ptr-A* with the sampling strategy [5]. Common parameters are selected for the genetic algorithm: The population size is the number of all possible hovering points in one instance, the maximal iteration is 10000, crossover is 0.1, and mutation probability is 0.8. The parameters of SA for the initial temperature, cooling coefficient, and maximal iteration are taken as 100, 0.99, and 1000, respectively.

3) Data Generation: We assume there is a probability distribution over a family of problems. During training, problem instances are generated according to this distribution, and any test examples are also produced from the same distribution at inference. For any problem instances, all CHs $\{b_1, \ldots, b_M\}$ are randomly sampled from the distribution $\boldsymbol{U} = \texttt{torch.FloatTensor}(1, 2).\texttt{uniform}(0, 3000)$. With the SNR threshold γ_{th} and environment parameters, we can calculate the hovering disk O_m for each CH b_m, and each cluster of candidate hovering points G_m is sampled from O_m. The number of nodes N_m in each ground cluster is randomly chosen from $\{5, 10, 15, 20, 25, 30\}$. Hence, any of the problem instances is obtained as $\boldsymbol{H}^{(\text{in})} = (c_0; (G_1, b_1, N_1); \ldots; (G_m, b_m, N_m); \ldots; (G_M, b_M, N_M))$.
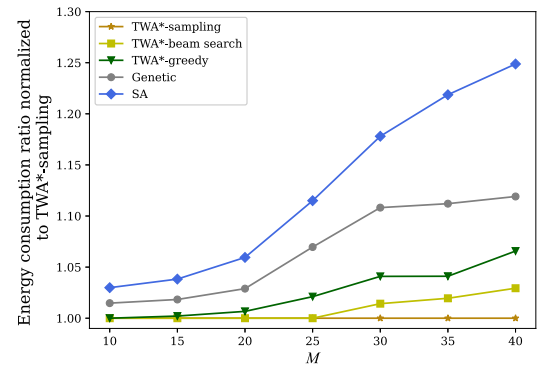
4) Environment Parameters and Hyperparameters: We consider a ground network with a size of 3 km $\times$ 3 km, and the start position of the UAV is located at $(0\,\text{m}, 0\,\text{m}, H\,\text{m})$. Environment parameters are listed in Table I. The embedding dimension d_{em} is equal to 512 and d_v is equal to 64. We train the proposed model using the Adam optimizer with a learning rate of 0.0001 on $E_{\text{epochs}} = 200$ epochs, where each epoch includes $S = 1000$ training steps. At each training step, the batch size B_{size} is equal to 512, which means there are 512 instances in each batch. In each instance, we set $M = 10$.

B. Analysis of the Results

We first compare the total AoI between our proposed algorithm against the genetic and SA algorithms on the trained model when the value of M varies. Although the model is trained on 10-clusters IoT networks ($M = 10$), it still shows good performance on IoT networks with different
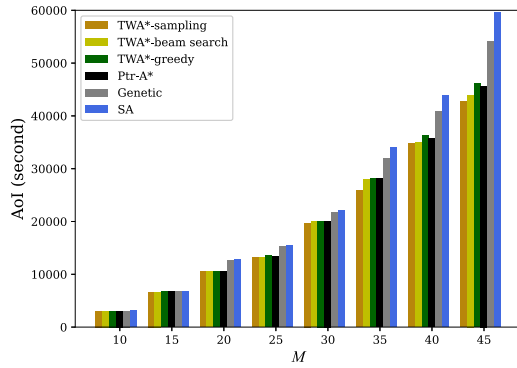
(a) Comparison of the total AoI when M varies.
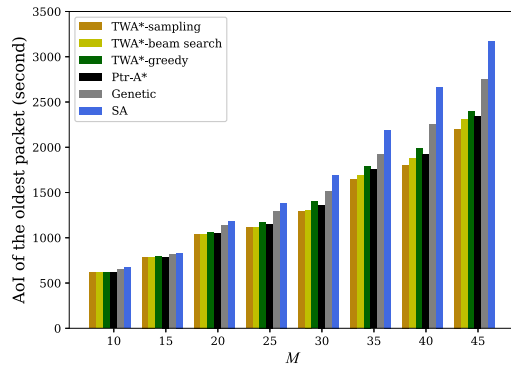
(b) Comparison of the oldest packet's AoI when M varies.

Fig. 5.   Comparison when M varies.

Fig. 6.   Comparison of energy consumption when M varies.

sizes, like 20-clusters, 30-clusters, etc., as can be seen in Fig. 5 (a). Specifically, the TWA*-sampling algorithm always obtains the minimal total AoI when compared with other two decoding methods, as well as the three other algorithms under comparison. The TWA*-beam search and TWA*-greedy algorithms exhibit an obviously superior performance than the genetic and SA algorithms in reducing the total AoI. The above observations indicate that the proposed algorithm with the three different decoding methods achieves an excellent generalization ability with respect to the size of the IoT network used for training. When $M = 10$, TWA*-sampling, TWA*-beam search, TWA*-greedy, Ptr-A*, and the genetic algorithms obtain almost the same total AoI; however, the SA algorithm has a higher total AoI when compared to our proposed algorithm with all three decoding strategies. As the value of M increases, the performance gap increases gradually between our proposed algorithm and comparison algorithms. For instance, when $M = 25$, the total AoI values of the TWA*-sampling, TWA*-beam search, TWA*-greedy, Ptr-A*, genetic, and SA algorithms are 13134, 13134, 13546, 13431, 15205, and 15452 seconds, respectively. Compared to Ptr-A*, TWA*-sampling has a performance gain of 2.2%. As M increases to 45, the total AoI values obtained by the TWA*-sampling, TWA*-beam search, and TWA*-greedy algorithms are 42803, 43971, and 46118 seconds, respectively. Compared to Ptr-A* with a AoI value of 45663 seconds, TWA*-sampling has a performance gain of 6.3%. However, the total AoI

values of the genetic and SA algorithms are 54061 and 59537 seconds, respectively, which are obviously inferior than what obtained by our proposed algorithm. In summary, our proposed algorithm using any of the three decoding methods can obtain better total AoI results than both the genetic and SA algorithms. In addition, TWA*-sampling always obtains better AoI values than Ptr-A* with the sampling strategy. This comparison result is consistent with the conclusions in [25] and [27] that the transformer-based technique outperforms the pointer network-based technique.

Next, we compare the AoI of the oldest packet which is from the node $b_{\pi(1)}^{(1)}$ that samples data first in the whole IoT network, among different algorithms. As can be seen in Fig. 5 (b), our proposed algorithm also exhibits a good performance in reducing the AoI of the oldest packet when compared with the genetic and SA algorithms. Furthermore, the TWA*-sampling algorithm obtains the best results among the three decoding methods and Ptr-A*.

Given that the proposed algorithm can find the best UAV trajectory with the minimal AoI in the UAV-IoT network among all the algorithms under comparison, it is of interest to further investigate the effective energy consumption of the UAV. It can be seen from (8) and (11) that the energy consumption of the UAV is related to its flying time and hovering time. The effective energy consumption is defined as the energy consumption of the UAV from the first visited hovering point to the end point, i.e., in completing its data collection task. Fig. 6 compares the effective energy consumptions for all the algorithms for different values of M. In particular, plotted in the figure are the average ratios of the effective energy consumptions by different algorithms with over that of the TWA*-sampling algorithm. As can be seen, our proposed algorithm with any of the three decoding methods has a better performance when compared to the genetic and SA algorithms, whereas the TWA*-sampling algorithm obtains the best result. The results in Fig. 6 are in line with expectations because with our proposed algorithm, the UAV spends less time to gather data than with the other two algorithms, which helps to reduce the effective energy consumption of the UAV.

Table II compares the running time at inference. As M increases, the running time of all the algorithms increases, which is well expected. We can see that among all the algorithms and for all the values of M, the running time of the

TABLE II

COMPARISON OF RUNNING TIME (SECOND)

| M	Algorithm					
	TWA*-sampling	TWA*-beam search	TWA*-greedy	Ptr-A*	Genetic	SA
10	1.9693	2.0653	1.9556	11.1556	47.57	5.5345
15	2.1412	2.2861	2.1055	19.3212	95.46	6.4262
20	2.3392	2.4900	2.3037	27.9023	163.41	6.8623
25	2.6006	2.8087	2.5778	36.9560	261.69	7.4619
30	2.8700	3.0876	2.8300	43.5498	392.97	8.3378
35	3.2018	3.4531	3.1536	52.4981	562.25	9.2576
40	3.8059	3.7506	3.5583	61.6301	749.16	9.6988
45	4.5112	4.5190	3.8995	75.8817	991.85	10.2019

TWA*-greedy algorithm is always shortest. Although TWA*-sampling obtains the best performance in reducing the total AoI as well as the AoI of the oldest packet (as can be seen from Fig. 5 (a) and Fig. 5 (b)), it has a longer running time than TWA*-greedy, which is reasonable. Similarly, TWA*-beam search takes slightly more time than TWA*-greedy because it needs more computational time to search for a better solution than TWA*-greedy. We can observe that the genetic algorithm takes the longest time among all the algorithms and its running time significantly increases as M increases. The running time of SA is acceptable in comparison with the genetic algorithm. Overall, the computational performance of our proposed model with all three decoding methods is significantly better than the SA and genetic algorithms. The running time of Ptr-A* is much greater than that of TWA*-sampling, TWA*-beam search, and TWA*-greedy. For example, when $M = 20$, the running time of the Ptr-A* is 11.9 times that of TWA*-sampling. When M increases to 45, the running time of the Ptr-A* is 16.8 times that of TWA*-sampling. This is because transformer-based techniques can process a sequence in parallel. However, with Ptr-A*, the elements of a sequence must be processed one by one. Hence, our proposed TWA* with three decoding methods is faster than Ptr-A*.

In order to provide insights about the effect of γ_{th} on the total AoI, we set the same number of devices, namely $N_m = 20$, in each ground cluster and evaluate in Fig. 7 (a) the performance of TWA*-sampling for different values of γ_{th}. According to *Lemma 1* and (6), the smaller the value of γ_{th} is, the larger the area of each hovering disk O_m will be. This will affect the positions of hovering points and thus the total AoI. As we can see in Fig. 7 (a), for any given number of ground clusters, the total AoI gradually increases as the value of γ_{th} decreases. For example, when $M = 30$, the values of total AoI in 10 dB, 20 dB, and 30 dB are 22707, 21655, and 20457 seconds, respectively. We can also observe that the total AoI gap among three values of γ_{th} increases as γ_{th} becomes higher.

Next, we compare the total flying time and the total hovering time of the UAV that make up of the AoI value $A_{\pi(1)}^{(1)}$ of the oldest packet in networks with different number of clusters. Specifically these total flying time and total hovering time of the UAV are calculated as $\sum_{t=1}^{M} T_{(c_{\pi(t)}, c_{\pi(t+1)})}^{(\text{fly})}$ and $\sum_{t=1}^{M} T_{c_{\pi(t)}}^{(\text{hov})}$, respectively. In each network, we also compare these portions of time when γ_{th} varies. Note that, the AoI values of the oldest packet are different for different γ_{th} values
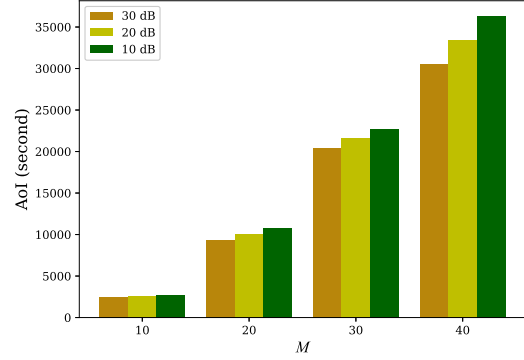
(a) Comparison of the total AoI for different values of γ_{th}.
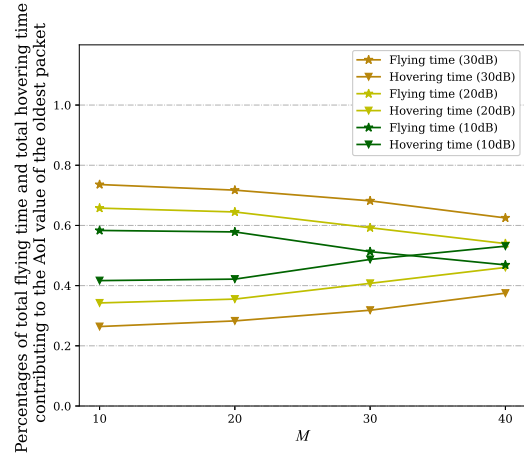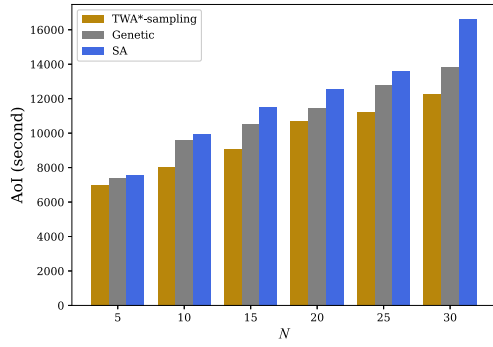

(b) Percentages of the total flying time and the total hovering time for different values of γ_{th}.

Fig. 7. Comparison for different values of γ_{th}.

in a network. The percentages of the total hovering and the total flying time that contribute to the AoI value of the oldest packet are plotted in Fig. 7 (b). When $M = 10$, the total flying time is always higher than the total hovering time for any thresholds γ_{th}. In addition, as γ_{th} increases, the flying time portion increases. This is because the selected hovering point may be closer to the center of each hovering disk if the value of γ_{th} is large, which will cause the flight distance to be longer and thus increases the total flying time. When M increases, the UAV needs more time to collect data, and we can see that the hovering time portion slowly increases as expected.

Fig. 8 (a) compares the total AoI for different algorithms and for different numbers of devices in each cluster. We test the

(a) Comparison of the total AoI when N varies
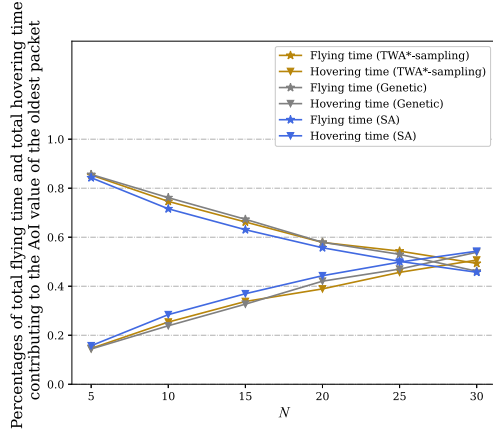
(b) Percentages of flying time and hovering time when N varies.

Fig. 8.   Comparison when N varies.

trained model on a 20-clusters instance with the same number of devices in each cluster. It can be seen that the proposed algorithm with the sampling decoding method always obtains the minimal values when compared to the genetic and SA algorithms. This clearly shows that our proposed algorithm can find a better trajectory in reducing the total AoI. As N increases, there is a large performance gap between TWA*-sampling and the two algorithms under comparison.

Fig. 8 (b) plots the percentages of the total flying time and the total hovering time that make up of the AoI value of the oldest packet in the 20-clusters network when N varies. For all the algorithms considered in Fig. 8 (b), as N increases, the percentage of the total hovering time gradually increases. This trend is justified because the UAV needs more time to collect data from a larger number of ground nodes.

V. Conclusion

In this paper, we have investigated and solved the problem of AoI-oriented data collection in UAV-enabled cluster-based IoT networks. With the aim of minimizing the total AoI of the collected data, we formulated the trajectory optimization problem as the GTSP by jointly optimizing the selection of hovering points of the UAV and the visiting order to these hovering points. To solve the formulated problem, we designed a novel algorithm framework based on the state-of-the-art transformer. In particular, the formulated trajectory planning problem is viewed as a "translation problem". The whole UAV-IoT network serves as the "source language" to the proposed model and the "target language" of the model is the UAV's trajectory with the minimal total AoI, where the transformer is utilized to generate the visiting order and the weighted A* is used to quickly find the hovering points. The proposed model is trained by reinforcement learning to learn a trajectory planning policy. Comprehensive experiments were conducted to evaluate the performance of the proposed algorithm. The obtained simulation results showed that the learned policy by the proposed algorithm has a strong gener-alization ability. When compared with other algorithms, our proposed algorithm with three different decoding methods not only reduces the total AoI, but also reduces the AoI of the oldest packet and the effective energy consumption of the UAV. Moreover, our method also has lower computation time. In future, we plan to extend the system model and the proposed algorithm to the multiple UAVs-assisted IoT network.

Appendix I
Proof of Lemma 1

By substituting (1)–(4) into (5), we can get the formulation (A.1), shown at the bottom of the page. For a fixed H, $20 \log_{10} \left(4\pi f_c \sqrt{H^2 + R^2_{(c_m, b_m)}} / c \right)$ is monotonically increasing with respect to $R_{(c_m, b_m)}$. As the analysis in [29] shows, $P^{(\text{LoS})}_{c_m}$ is monotonically increasing with respect to θ_{c_m}. Since $\theta_{c_m} = \arctan(H / R_{(c_m, b_m)})$, $P^{(\text{LoS})}_{c_m}$ is monotonically decreasing with respect to $R_{(c_m, b_m)}$ for a fixed H. Then, $(\xi_{\text{LoS}} - \xi_{\text{NLoS}}) \Big/ \left(1 + \beta \exp \left(-\widetilde{\beta} \left(\theta_{c_m} - \beta \right) \right) \right)$ is monotoni-cally increasing with respect to $R_{(c_m, b_m)}$ because $\xi_{\text{LoS}} < \xi_{\text{NLoS}}$. Finally, we arrive at the conclusion that the left side of (A.1) is monotonically decreasing with respect to $R_{(c_m, b_m)}$ for a fixed H. When the SNR γ_{c_m} decreases to the threshold γ_{th}, we can obtain the maximum R^*. Hence, for any c_m, the UAV can successfully receive data from b_m if $R_{(c_m, b_m)} \leq R^*$.

$$P_{\text{CH}} \frac{1}{\sigma^2 \left(P^{(\text{LoS})}_{c_m} L^{(\text{LoS})}_{c_m} + \left(1 - P^{(\text{LoS})}_{c_m} \right) L^{(\text{NLoS})}_{c_m} \right)} \geq \gamma_{\text{th}},$$

$$P_{\text{CH}} \frac{1}{\sigma^2 \left(20 \log_{10} \left(\frac{4\pi f_c d_{(c_m, b_m)}}{v_{\text{light}}} \right) + P^{(\text{LoS})}_{c_m} (\xi_{\text{LoS}} - \xi_{\text{NLoS}}) + \xi_{\text{NLoS}} \right)} \geq \gamma_{\text{th}},$$

$$P_{\text{CH}} \frac{1}{\sigma^2 \left(20 \log_{10} \left(\frac{4\pi f_c \sqrt{H^2 + R^2_{(c_m, b_m)}}}{v_{\text{light}}} \right) + \frac{\xi_{\text{LoS}} - \xi_{\text{NLoS}}}{1 + \beta \exp \left(-\widetilde{\beta} (\theta_{c_m} - \beta) \right)} + \xi_{\text{NLoS}} \right)} \geq \gamma_{\text{th}}. \qquad \text{(A.1)}$$

REFERENCES

[1] S. Hu, X. Chen, W. Ni, E. Hossain, and X. Wang, "Distributed machine learning for wireless communication networks: Techniques, architectures, and applications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1458–1493, 3rd Quart., 2021.

[2] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[3] S. Hu, W. Ni, X. Wang, A. Jamalipour, and D. Ta, "Joint optimization of trajectory, propulsion, and thrust powers for covert UAV-on-UAV video tracking and surveillance," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1959–1972, 2021.

[4] S. Hu, Q. Wu, and X. Wang, "Energy management and trajectory optimization for UAV-enabled legitimate monitoring systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 142–155, Jan. 2021.

[5] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9540–9554, Sep. 2021.

[6] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular internet of UAVs: Sensing and communication trade-off design," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6578–6592, Oct. 2020.

[7] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 2731–2735.

[8] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1211–1223, Jan. 2021.

[9] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for UAV-assisted data collection," in *Proc. IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM)*, Apr. 2018, pp. 553–558.

[10] J. Liu, P. Tong, X. Wang, B. Bai, and H. Dai, "UAV-aided data collection for information freshness in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2368–2382, Apr. 2021.

[11] Z. Jia, X. Qin, Z. Wang, and B. Liu, "Age-based path planning and data acquisition in UAV-assisted IoT networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[12] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in UAV-assisted IoT networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 2003–2008, Feb. 2019.

[13] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5994–6006, Sep. 2021.

[14] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *Proc. IEEE INFOCOM Workshops*, Jul. 2020, pp. 716–721.

[15] D. Ebrahimi, S. Sharafeddine, P.-H. Ho, and C. Assi, "UAV-aided projection-based compressive data gathering in wireless sensor networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1893–1905, Apr. 2019.

[16] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten, "Energy minimization in UAV-aided networks: Actor-critic learning for constrained scheduling optimization," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5028–5042, May 2021.

[17] L. Shen, N. Wang, Z. Zhu, Y. Fan, X. Ji, and X. Mu, "UAV-enabled data collection for mMTC networks: AEM modeling and energy-efficient trajectory design," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[18] J. T. Isaacs and J. P. Hespanha, "Dubins traveling salesman problem with neighborhoods: A graph-based approach," *Algorithms*, vol. 6, no. 1, pp. 84–99, Feb. 2013.

[19] R. Penicka, J. Faigl, P. Vana, and M. Saska, "Dubins orienteering problem with neighborhoods," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 1555–1562.

[20] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing Age-of-Information in UAV-assisted networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[21] W. Li, L. Wang, and A. Fei, "Minimizing packet expiration loss with path planning in UAV-assisted data sensing," *IEEE Wireless Commun. Lett.*, vol. 8, no. 6, pp. 1520–1523, Dec. 2019.

[22] A. Ferdowsi, M. A. Abd-Elmagid, W. Saad, and H. S. Dhillon, "Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1250–1265, May 2021.

[23] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Jun. 2017, pp. 5998–6008.

[24] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in *Proc. Int. Conf. Integr. Constraint Program., Artif. Intell., Oper. Res. (CPAIOR)*, Jun. 2018, pp. 170–181.

[25] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2019, pp. 1–25.

[26] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 1–13, Apr. 2021.

[27] X. Bresson and T. Laurent, "The transformer network for the traveling salesman problem," 2021, *arXiv:2103.03012*.

[28] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[29] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[30] J. Yao and N. Ansari, "QoS-aware power control in Internet of Drones for data collection service," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6649–6656, Jul. 2019.

[31] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.

[32] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "UAV-enabled age-optimal data collection in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, May 2019, pp. 1–6.

[33] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *J. Algorithms*, vol. 48, no. 1, pp. 135–159, Aug. 2003.

[34] 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.

[35] Concorde. [Online]. Available: http://www.math.uwaterloo.ca/tsp/concorde.html

[36] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," 2021, *arXiv:2106.04554*.

[37] R. Ebendt and R. Drechsler, "Weighted A search unifying view and application," *Artif. Intell.*, vol. 173, no. 14, pp. 1310–1342, Sep. 2009.

[38] G. Ramalingam and T. Reps, "On the computational complexity of dynamic graph problems," *Theor. Comput. Sci.*, vol. 158, nos. 1–2, pp. 233–277, May 1996.

[39] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[40] J. Yang, C. Wu, H. P. Lee, and Y. Liang, "Solving traveling salesman problems using generalized chromosome genetic algorithm," *Prog. Natural Sci.*, vol. 18, no. 7, pp. 887–892, Jul. 2008.

[41] S.-H. Zhan, J. Lin, Z.-J. Zhang, and Y.-W. Zhong, "List-based simulated annealing algorithm for traveling salesman problem," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–12, Mar. 2016.

Botao Zhu is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, Canada. His current research interests include wireless sensor networks, the Internet-of-Things, machine learning, and unmanned aerial vehicles.

Ebrahim Bedeer (Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees from Tanta University, Tanta, Egypt, and the Ph.D. degree from Memorial University, St. Johns, NL, Canada, all in electrical engineering.

In 2019, he joined the Department of Electrical and Computer Engineering, University of Saskatchewan as an Assistant Professor. Before that, he was an Assistant Professor (Lecturer in the U.K.) at Ulster University, U.K., and a Postdoctoral Fellow at Carleton University, Ottawa, ON, Canada, and The University of British Columbia, Kelowna, BC, Canada. His current research interests include applications of optimization techniques in signal processing and wireless communications, spectral efficient communication systems, and the Internet-of-Things (IoT). He is a Registered Member of the Association of Professional Engineers and Geoscientists of Saskatchewan (APEGS).

Robert Barton (Member, IEEE) received the degree in engineering physics from The University of British Columbia. He is currently a Distinguished Engineer at the Cisco's Digital Transformation and Innovation Group, where he has the role of the Chief Architect for Cisco Canada and Cisco's Global IoT Sales Organization. He has worked in the IT industry for over 23 years, the last 20 of which have been at Cisco. He is a published author, with titles on the subjects of network QoS, wireless, the IoT, machine learning, and data analytics. He has also contributed to many academic papers, and leads Cisco's university research partnership program. He also holds many patents in the areas of wireless communications, segment routing, and machine learning. His current areas of work include wireless communications, industrial networking, the IoT, and AI/ML in networking systems.

Ha H. Nguyen (Senior Member, IEEE) received the B.Eng. degree from the Hanoi University of Technology (HUT), Hanoi, Vietnam, in 1995, the M.Eng. degree from the Asian Institute of Technology (AIT), Bangkok, Thailand, in 1997, and the Ph.D. degree from the University of Manitoba, Winnipeg, MB, Canada, in 2001, all in electrical engineering. He joined the Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada, in 2001, and became a Full Professor in 2007. He currently holds the position of NSERC/Cisco Industrial Research Chair in low-power wireless access for sensor networks. He is the coauthor (with Ed Shwedyk) of the textbook *A First Course in Digital Communications* (published by Cambridge University Press). His research interests fall into broad areas of communication theory, wireless communications, and statistical signal processing. He is a fellow of the Engineering Institute of Canada (EIC) and a Registered Member of the Association of Professional Engineers and Geoscientists of Saskatchewan (APEGS). He served as a technical program chair for numerous IEEE events. He was an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE WIRELESS COMMUNICATIONS LETTERS from 2007 to 2011 and from 2011 to 2016. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Zhen Gao (Member, IEEE) received the B.S. degree in information engineering from the Beijing Institute of Technology, Beijing, China, in 2011, and the Ph.D. degree in communication and signal processing with the Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, in 2016. He is currently an Associate Professor with the Beijing Institute of Technology. His research interests are in wireless communications, with a focus on multicarrier modulations, multiple antenna systems, and sparse signal processing. He was a recipient of the IEEE Broadcast Technology Society 2016 Scott Helt Memorial Award (Best Paper), the Exemplary Reviewer of IEEE COMMUNICATION LETTERS in 2016, *IET Electronics Letters* Premium Award (Best Paper) in 2016, and the Young Elite Scientists Sponsorship Program (2018–2021) from the China Association for Science and Technology.