ORIGINAL ARTICLE

UAV parcel Delivery System with Deep Reinforcement Learning Based Collision Avoidance and Route Optimization

Chun-Yuan Chi[1], De-Fu Chen[2], Hoang-Phuong Doan[2], and Chung-Hsien Kuo[2, *]

[1]*Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan*
[2] *Department of Mechanical Engineering, National Taiwan University, Taipei, Taiwan*
***Corresponding author:** chunghsien@ntu.edu.tw

(**Received** 28 June 2024, **Accepted** 09 July 2024 , **Published Online** 16 July 2024)
This work was supported by the National Science and Technology Council, Taiwan under the Grant NSTC 112-2221-E-011-140. (**Corresponding author:** Chung-Hsien Kuo).
DOI: https://doi.org/10.5875/p25fht10

Abstract: Unmanned Aerial Vehicles (UAVs), or drones, have recently become a favorable solution for fast parcel delivery due to their maneuverability and advances in navigation technologies. With the limitation of battery capacity and payload of drones, it is crucial to consider both efficiency and cost while conducting the tasks. Meanwhile, UAVs should not collide with each other while traveling to customers. In this article, we propose a UAV parcel delivery system involving deep reinforcement learning (DRL) approach for collision avoidance and a genetic algorithm for route optimization. Specifically, a delivery center generates near-optimal routes, loading UAV with parcels according to demands. Each UAV takes charge of delivering packages in compliance with the assigned route while avoiding collision with each other. We utilize DRL to achieve collision avoidance without having prior knowledge about the trajectories of other UAVs. Additionally, we adopt a genetic algorithm to obtain the lowest energy cost path for each UAV. To find such an optimized path, we solve a capacitated vehicle routing problem (CVRP) with a modified cost function and extra constraints. Realistic simulations using a physics engine and software-in-the-loop (SITL) are conducted to evaluate the feasibility of the proposed methods.

Keywords: Capacitated vehicle routing problem, Genetic Algorithm, Reinforcement Learning, UAV collision avoidance.

I. Introduction

Unmanned Aerial Vehicles (UAVs) are recently being explored due to their potential to be adopted in parcel delivery systems. With the growth of demands in the logistics and e-commerce industry, many logistics companies have shown great interest in using UAVs for delivery since parcels can be delivered more efficiently. Amazon Prime Air has become a pioneer in this field, expecting to adopt drones to replace delivery trucks, making the delivery faster compared to the present day [1]. This parcel delivery scheme is called the drone-only scheme, where the drone departs from the depot, delivers goods to the customer, and then returns to the depot. This scheme suffers from the distance limitation due to the battery capacity of the drone. Several publications proposed methods to expand the coverage area by adding charge stations and warehouses in the network [2], [3], [4].

In recent years, another scheme, called the Truck-and-Drone scheme or last-mile drone delivery, has drawn much attention in the research community. The Truck-and-Drone scheme expands the delivery coverage and improves energy efficiency by combining heterogeneous vehicles. To coordinate drones with trucks, Wang et al. proposed a heuristic routing and scheduling algorithm to solve the hybrid parcel delivery problem [5]. Nirupam Das et al. synchronized drones and delivery trucks by developing a multi-objective optimization model that minimizes travel costs and maximizes customer service level in terms of timely deliveries [6]. Chen et al. developed a drone delivery system that achieves mixed indoor- outdoor autopilot operation [7].

For a system with multiple UAVs, it is crucial to consider the risk of collision among them. It is difficult for each UAV to avoid others without a centralized control mechanism. Therefore, we propose a decentralized collision avoidance method by adopting a DRL-based approach. Additionally, each UAV has a payload limit which limits the weight and quantity of parcels the drone can carry. The sequence of how UAVs travel to customers becomes critical since the gross weight of the UAV will directly impact battery drain and, furthermore, affect the logistics cost. As a solution, we aim to minimize the total energy consumed by UAVs while traveling instead of minimizing total travel distance like the traditional capacitated vehicle routing problem (CVRP). For optimization, we adopt a genetic algorithm with a custom fitness function.

In this article, we demonstrate the feasibility of the proposed collision avoidance method by conducting realistic simulations in a physics engine with SITL flight controller simulation on Robot Operating System (ROS) [8]. We also simulate the entire UAV parcel delivery process in a campus environ- ment, including loading packages, traveling to customers, and dropping packages. The contributions of this article can be summarized as follows:

- An application of DRL-based obstacle avoidance method to a parcel delivery system consisting of multiple UAVs with Soft Actor Critic Framework.
- The usage of a genetic algorithm to generate suitable routes for multiple UAVs in the system based on total energy consumption.

The rest of this article is organized as follows: Section II presents related work. Section III states the system architecture and the problem definitions. Section IV addresses the UAV collision avoidance problem with a state-of-the-art deep reinforcement learning algorithm. Section V discusses the route optimization problem for capacitated vehicles in UAV parcel delivery tasks. Section VI carries out the validation and simulation results. Section VII concludes the article.

II. Related Works

A. UAV Collision Avoidance

Collision avoidance is a fundamental requirement for multi-UAV systems. Centralized collision models for UAVs are presented. Loayza et al. proposed a centralized model providing feedback in real-time to the agents while considering trajectory calculation and collision avoidance [9]. Mellinger et al. presented an algorithm for the generation of optimal trajectories for teams of heterogeneous quadrotors in three-dimensional environments with obstacles by formulating the problem us- ing mixed-integer quadratic programs (MIQPs) [10]. Both solutions rely on a central server communicating with every agent and generating global control commands according to the observations for all UAVs. However, implementing such centralized systems in large environments is usually difficult since it heavily relies on the communication between agents and the server; delay or interference in signal transmissions may lead to unwanted results.

In recent years, researchers have turned their interest into decentralized methods. In such systems, agents take action based only on their own observations. Several works [11]– [13] have successfully adopted reinforcement learning to train policies to plan collision-free trajectories by leveraging local observations. However, the method proposed in [11] assumes all UAVs fly at the same speed and can only output dis- cretized turning angles, which may cause jerky movement. The off-policy actor-critic-based reinforcement learning algorithm, deep deterministic policy gradient (DDPG) [14], used in [12], is hyperparameter-sensitive [15] and suffers from finding the optimal policy due to its non-stochastic characteristic. Researchers in [13] formulated the UAV collision avoidance problem with wireless connectivity constraints as a Markov decision process (MDP) and optimized the value function of the MDP to find the optimal policy. However, the proposed method was not tested in a realistic environment, so one cannot guarantee the feasibility of the policy in an environment with noises or delays.

A method based on Multi-Agent Reinforcement Learn- ing is presented in [16] to conduct large-scale searching in an unknown environment with multi-UAVs. Furthermore, a new multi-agent recurrent deterministic policy gradient (MARDPG) algorithm is proposed in [17] to achieve the goal of obstacle avoidance for multi-UAVs. In [18], a Deep Reinforcement Learning-based collision avoidance algorithm with Attention-Based Policy Distillation is proposed, enabling UAVs to conduct obstacle avoidance more efficiently and accurately. Hui et al. discuss the use of a Decentralized Exploration Planning approach based on a lightweight in- formation structure for multi-UAV systems [19]. Although various approaches for tackling collisions among UAVs have been published, none were applied to parcel delivery systems, indicating significant potential for research in this field.

B. Scheduling and Routing Problems for Drone Delivery

As drone technology matures, many large organizations have shown interest in drone delivery. Even though significant efforts have been put into developing drone delivery tech- nologies, the drone delivery planning problem poses a new challenge due to limited flight range and payload of drones. Traditional traveling salesman problems (TSP) or vehicle rout- ing problems (VRP) are no longer adequate for formulating the drone delivery routing problem. Some variants, such as [20] and [21], proposed flying sidekick traveling salesman problem (FSTSP) and vehicle routing problem with drones (VRPD) respectively. The VRPD is considered an extension of the FSTSP. While the FSTSP considers only one drone and one truck in the entire operation, VRPD utilizes multiple trucks and drones to make deliveries while considering the capacity of both trucks and drones. However, both works ignored factors crucial to practical drone delivery, such as changing payload weights and energy costs. Yao et al. discovered a scheduling approach using an Evolutionary Utility Prediction Matrix, which can adapt to the environment dynamically [22]. However, precise tuning of parameters is needed.

Recently, several researchers have focused on reducing cost or energy consumption. Dorling et al. proposed solving drone delivery problems (DDPs) with a multi-trip VRP (MTVRP) [23]. They focused on minimizing cost or delivery time while considering battery weight, payload weight, and drone reuse. In [24], the authors focused on minimizing the total energy consumption in electric vehicle routing problems with drones (EVRPD). In this thesis, we consider a similar scenario with drones being the only vehicle in the system. We adopt the energy cost function in [24] and use it in a genetic algorithm.

III. Problem Definitions

We address two major problems in this article: UAV col- lision avoidance and optimized route solutions in the parcel delivery system. We first state the definitions of the two problems respectively, then delve into detailed content in the following sections.

A. UAV Collision Avoidance

Consider an environment consisting of a set $\mathcal{M} = \{1,\ 2,\ \ldots,\ m\}$ of m UAVs flying at a constant altitude. For each UAV to avoid collisions with others we must ensure the distance $d_{i,j}$ between each UAV is less than the minimum collision distance d_{min} as shown in (1).

$$d_{i,j} < d_{min} \ \ \forall i,j \in \mathcal{M}, i \neq j \tag{1}$$

In this article, we also assume a UAV k can fly with arbitrarily speed v_k and heading angle θ_k where v_k and θ_k are continuous values instead of discrete values. Our goal here is to find an approach to make UAVs able to avoid each other or obstacle automatically, and the approach we adopt is Deep Reinforcement Learning (DRL)

B. Routes Optimization

We consider a capacitated vehicle routing problem (CVRP) with both weight and volume constraints since UAVs have a limited payload and space to carry parcels. Instead of minimizing the traveling cost directly calculated with the traveled distance of vehicles in conventional CVRP, we aim to reduce the total energy cost since modern UAVs use batteries as power sources and the electricity consumption directly affects logistics costs. We define our energy cost function of UAV as follows, similar to [24].

$$Cost_{Energy} = Distance \times (1 + Payload) \tag{2}$$

With the definition of cost function (2), our goal is to find a solution that minimize the cost function. In this article, we apply genetic algorithm to solve the optimization problem.

IV. DRL BASED UAV COLLISION AVOIDANCE

We propose an approach for UAV collision avoidance using a state-of-the-art deep reinforcement learning algorithm called Soft Actor-Critic (SAC) [25]. Specifically, SAC is an off-policy actor-critic deep reinforcement learning algorithm based on the maximum entropy reinforcement learning framework. Unlike Deep Deterministic Policy Gradient (DDPG) [26] or Twin Delayed Deep Deterministic Policy Gradient (TD3) [27], SAC uses a stochastic policy with entropy regularization instead of a deterministic policy.

A. State Space

Since method proposed only consider local observation of the agent, the state representation would be represented in the local coordinate system of UAV. Consider a tagged UAV agent in Fig. 1, the x-axis is the current heading of the agent, d_{goal} is the relative distance of the agent's goal, d_{obs} is the relative distance of between the detected obstacle UAV and the tagged agent, θ_{goal} is the angle between the agent's goal and the x-axis, θ_{obs} is the angle between the detected obstacle and the x-axis, ψ_{obs} is the heading of the obstacle UAV with respect to the x-axis. It is assumed that all UAVs fly toward their heading direction and all the angles and headings are within the range between $-\pi$ and $+\pi$.

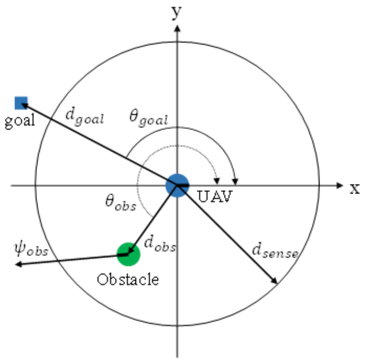


Fig. 1. The local state representation of a UAV agent.

Let $\mathbf{s}_t$ be the current observed state of the tagged UAV at time t where $t \in [0, \infty)$. $\mathbf{s}_t$ is a composition of agent information info_{agent} and two nearest obstacle information info_{obs1} and info_{obs2}.

$$\mathbf{s}_t = [\text{info}_{agent} \quad \text{info}_{obs1} \quad \text{info}_{obs2}] \tag{3}$$

where info_{agent} includes the information of the current agent speed v_{agent}, current heading angle ψ_{agent} with respect to the world coordinate system, relative goal distance dgoal, and θ_{goal} the angle between the agent's goal and the x-axis. To keep state values in the same order of magnitude, each term in info_{agent} is divided by a constant denominator where v_{max} is the maximum velocity of all UAVs in the system and d_{scale} is the predefined distance normalization factor.

$$\text{info}_{agent} = \begin{bmatrix} \frac{v_{agent}}{v_{max}} & \frac{\psi_{agent}}{\pi} & \min\left(1, \frac{d_{goal}}{d_{scale}}\right) & \frac{\theta_{goal}}{\pi} \end{bmatrix} \tag{4}$$

info_{obs} represents the information of the obstacle UAV, including relative distance d_{obs}, relative angle θ_{obs} and ψ_{obs} the heading of the obstacle UAV. Likewise, each term in info_{obs} is divided by a constant denominator.

$$\text{info}_{\text{obs}} = \begin{bmatrix} \frac{d_{obs}}{d_{sense}} & \frac{\theta_{obs}}{\pi} & \frac{\psi_{obs}}{\pi} \end{bmatrix} \tag{5}$$

It is worth noticing that $\text{info}_{obs} = [1 \quad 1 \quad 0]$ if the tagged UAV cannot detect any UAV at the time.

B. Action Space

We model the quadrotor UAVs as a unicycle model as in Fig. 2. The velocity of a UAV moving in the x-axis is given by $v_x = v\cos\theta$ and the velocity of a UAV moving in the y-axis is given by $v_y = v\sin\theta$. Let $\mathbf{a}_t$ denotes the action space, $v \in [0, v_{max}]$ be the UAV flying velocity and $\theta \in [-\pi, \pi]$ be the heading angle of the UAV with respect to the world coordinate system.

$$\mathbf{a}_t = \begin{bmatrix} \frac{2v}{v_{max}} - 1 & \frac{\theta}{\pi} \end{bmatrix} \tag{6}$$

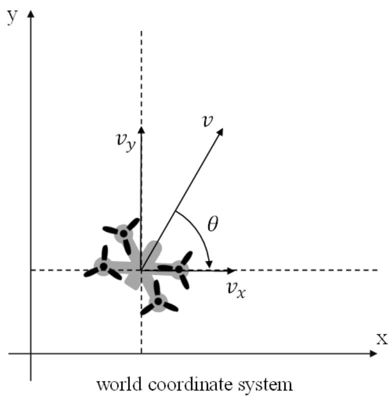


Fig. 2. Unicycle model of a quadrotor UAV.

Since the last layer of the actor network is activated by a hyperbolic tangent function, we resize and normalized the action space in order to fully utilize the numeric range of the action space.

C. Reward Function Design

We adopt the design of reward function in [11] and added an addition penalty value which is described in the next paragraph, and the algorithm is shown in Algorithm 1.

```
Algorithm 1 REWARD FUNCTION $\mathcal{R}$
Input: $d_{goal,t}$ $d_{goal,t+1}$ $\theta_{goal,t+1}$ $d_{obs,t+1}$ $d_{init}$ $v_{max}$ $d_{colli}$ $d_{min}$
Output: $r_t$
  if $d_{obs,t+1} \leq d_{colli}$ then
    $r_t \leftarrow -2$
  else
    if $r_t > 0$ then
      $r_t \leftarrow r_t\left(1 - \frac{d_{goal,t+1}}{1.5 d_{init}}\right)$
    else
      $r_t \leftarrow r_t\left(1 + \frac{d_{goal,t+1}}{1.5 d_{init}}\right)$
    end if
    $r_t \leftarrow r_t - 0.01|\theta_{goal,t+1}|$
    $r_t \leftarrow r_t - 0.01 \min\left(\frac{v_{max}}{d_{init}}, 1\right)$
  end if
```

Let r_t denotes the reward agent receives at time $t \in [0, \infty)$ and $\mathcal{R}: \mathcal{S} \rightarrow r_t$ be the reward function where $\mathcal{S}$ is the global state of the environment. Let d_{colli} be the collision threshold for reward calculation and $d_{obs,t+1}$ be the relative obstacle distance of the new state $\mathbf{s}_{t+1}$. If $d_{obs,t+1} < d_{colli}$, the agent is punished by a reward value $r_t = -2$ to help it learn how to avoid collisions. Otherwise, r_t is initialized with a value proportional to the relative velocity between the agent and its goal. The reward is positive as the agent moves towards its goal and negative as the agent moves away from its goal.

The initial reward value is also scaled by a factor to make the reward value larger as the agent approaches the target even if the speed of the agent remains unchanged. $|\theta_{goal,t+1}|$ is then subtracted from r_t after multiplied with a small factor. This helps the agent learns to navigate to the goal by applying punishments when it is not flying in the direction of the goal. Finally, an addition value inversely proportional to the initial goal distance d_{init} is subtract from r_t to encourage the agent to learn to reach its target as soon as possible.

D. Soft Actor Critic Framework in UAV Collision Avoidance

Conventional reinforcement learning aims to learn a policy $\pi(\mathbf{a}_t, \mathbf{s}_t)$ that maximize the expected sum of rewards $\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}[r(\mathbf{s}_t, \mathbf{a}_t)]$ where ρ_π denotes the state action distribution of policy π and $\mathbb{E}$ denotes the expectation value function. However, SAC generalizes the standard objective by augmenting it with an entropy term $\mathcal{H}(P) = \mathbb{E}_{x\sim P}[-\log P(x)]$where x is a random variable with probability density function P with respect to x, such that the optimal policy π^* additionally aims to maximize its entropy at each visited state [28]:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha\mathcal{H}(\pi(\cdot\,|\mathbf{s}_t))] \tag{7}$$

where $\alpha > 0$ is the temperature parameter that determines the relative importance of the entropy term versus the reward $r(\mathbf{s}_t, \mathbf{a}_t)$. The concept of the implemented entropy term $\mathcal{H}(\pi(\cdot\,|\mathbf{s}_t))$ represents the randomness of the outputs of the stochastic policy π with given $\mathbf{s}_t$. The extra entropy term incentivizes the policy to explore more widely, improving its robustness against perturbations [29].

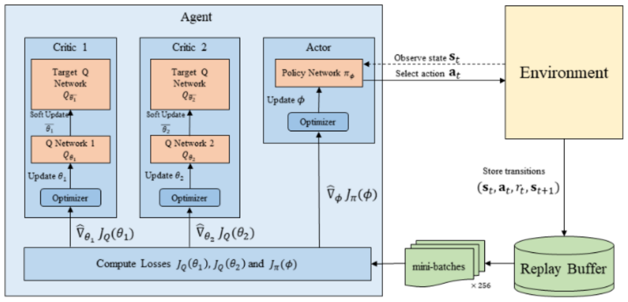

Agent
Critic 1
Target Q Network $Q_{\bar{\theta}_1}$
Soft Update
$\bar{\theta}_1$
Q Network 1 Q_{θ_1}
Update θ_1
Optimizer
Critic 2
Target Q Network $Q_{\bar{\theta}_2}$
Soft Update
$\bar{\theta}_2$
Q Network 2 Q_{θ_2}
Update θ_2
Optimizer
Actor
Policy Network π_ϕ
Update ϕ
Optimizer
Observe state $\mathbf{s}_t$
Select action $\mathbf{a}_t$
Environment
Store transitions $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$
$\hat{\nabla}_\phi J_\pi(\phi)$
$\hat{\nabla}_{\theta_1} J_Q(\theta_1)$
$\hat{\nabla}_{\theta_2} J_Q(\theta_2)$
Compute Losses $J_Q(\theta_1), J_Q(\theta_2)$ and $J_\pi(\phi)$
mini-batches
× 256
Replay Buffer

Fig. 3. An illustration of a Soft Actor Critic framework.

The SAC deep reinforcement learning algorithm framework is shown in Fig. 3. It is composed of five neural networks: A Gaussian policy network π_ϕ with parameters ϕ, two Q-networks Q_{θ_1} and Q_{θ_2} with parameters θ_1 and θ_2, and two target Q-networks $Q_{\bar{\theta}_1}$ and $Q_{\bar{\theta}_2}$ with parameters $\bar{\theta}_1$ and $\bar{\theta}_2$. In order to train the deep reinforcement learning model, we must first compute the loss function of each network in the SAC framework. As suggested in [28], we learn the Q-network parameters as a regression problem by minimizing the following loss function:

$$J_Q(\theta_i) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\mathcal{D}}\left[\left(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V_{\bar{\theta}_1,\bar{\theta}_2}(\mathbf{s}_{t+1})\right)\right)^2\right] \tag{8}$$

using mini-batches from the replay buffer $\mathcal{D}$, where the value function $V_{\bar{\theta}_1,\bar{\theta}_2}$ is implicitly defined through the Q-networks and the policy as:

$$V_{\bar{\theta}_1,\bar{\theta}_2}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\left[\min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right] \tag{9}$$

Then, we improve the Gaussian policy in a similar factor by minimizing:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\mathbf{a}_t\sim\pi}\left[\alpha \log \pi(\mathbf{a}_t|\mathbf{s}_t) - \min_{i\in\{1,2\}} Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{10}$$

using the reparameterization trick. We re-parameterize the policy function using a neural network transformation of $\mathbf{a}_t = f_\phi(\varepsilon_t|\mathbf{s}_t)$ where ε_t is an input noise vector sampled from a normal Gaussian distribution as suggested in [28]. Furthermore in [28], the author proposed a way to automate the process of choosing the optimal temperature α. Instead of requiring the user to set the temperature manually, they automate this process by formulating a different maximum entropy reinforcement learning objective:

$$J(\alpha) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\mathbf{a}_t\sim\pi_\phi}[-\alpha \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - \alpha\bar{\mathcal{H}}] \tag{11}$$

where $\bar{\mathcal{H}}$ denote the target entropy and is set equal to the dimension of the action space. As in [28] we select target entropy as the dimension of the action space, letting $\bar{\mathcal{H}} = -\dim(\mathbf{a}_t) = -2$. Finally, the target Q-networks $Q_{\bar{\theta}_i}, i \in \{1,2\}$ are updated with a delay factor τ with respect to the original Q-networks as shown in (12).

$$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i, i \in \{1,2\} \tag{12}$$

Algorithm 2 ALGORITHM OF SOFT ACTOR-CRITIC WITH AUTOMATIC ENTROPY ADJUSTMENT

```
Initialize network parameters $\theta_1, \theta_2, \overline{\theta}_1, \overline{\theta}_2, \phi$ and entropy temperature coefficient $\alpha$
foreach episode do
    foreach environment step do
        foreach angent do
            $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
            $\mathbf{s}_{t+1}, r_t \sim environment(\mathbf{s}_{t+1}, r_t|\mathbf{s}_t, \mathbf{a}_t)$
            $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}\}$
        end
        $\theta_i \leftarrow \theta_i - \lambda \hat{\nabla}_\theta J_Q(\theta_i)$ for $i \in \{1, 2\}$
        $\phi \leftarrow \phi - \lambda \hat{\nabla}_\phi J_\pi(\phi)$
        $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J_\pi(\alpha)$
        $\overline{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\overline{\theta}_i$ for $i \in \{1, 2\}$
    end
end
```

While training, all objectives in Equations (8), (10), and (11) are all optimized simultaneously. Algorithm 2 summarizes the full training procedure, where $\hat{\nabla}$denotes stochastic gradients and λ denotes the learning rate. The parameters are first initialized. During environment steps of each iteration, the algorithm will sample action, $\mathbf{a}_t$, from action space, $\pi_\varphi(\mathbf{a}_t|\mathbf{s}_t)$, and transition state, $\mathbf{s}_{s+1}$, from the environment regarding current environment state, $\mathbf{s}_t$ and action taken, $\mathbf{a}_t$. These data with the reward $\boldsymbol{r}_t$, will then be stored in the replay buffer, $\mathcal{D}$ for each agent. The parameters then will be optimized using gradient descend on the basis of equation (8), (10), (11) and (12).

E. Training Environment

To increase training efficiency, instead of using a physics engine, we create a custom OpenAI Gym [30] environment for developing and testing learning agents written in Python. In each episode, the environment is initialized by generating M agents with randomly distributed depots $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ and goals $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, where $\forall i, j \in \mathcal{M}$ and i ≠ j $G_i \neq G_j$. At each time slot t the environment takes m sets of actions $\mathbf{a}_t$ and returns m sets of new states $\mathbf{s}_{t+1}$ and rewards r_t with respect to each agent m. The goal of the environment is to let the model learn a single policy over all agents while all agents can successfully reach their targets and avoid collisions. Each episode of training is terminated either when the agent reaches its goal or when the agent flies out of the boundary. Each episode will end only when all agents have reached their goals or time has run out.

V. ROUTE OPTIMIZATION IN UAV PARCEL DELIVERY TASKS

A. Modified CVRP

We discuss the solution to the route optimization in UAV delivery system in this section. we will formulate the modified CVRP (mCVRP) with the proposed energy cost function in (2), which is consider an extension of the conventional CVRP problem.

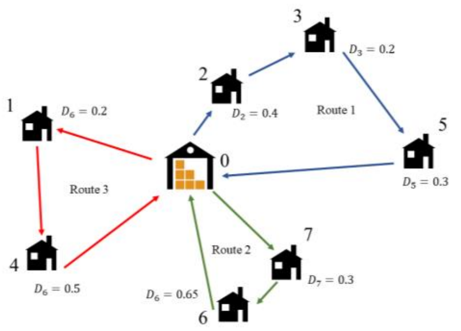


Fig. 4. Example of a solution of the mCVRP.

Consider an environment with a total number of n nodes and $n-1$ customers as illustrated in Fig. 4., we denote the location of each node as L_i where $i \in \{1,2,\ldots,n\}$. L_0 is defined as the location of the depot while $L_{i,i\neq 0}$ is the location of the customers, the distance between node i and node j is represented by $d_{i,j}$. It is assumed that each UAV is able to carry at most C_p parcels at once and have a payload limit of C_w. The solution space of the proposed mCVRP is composed of three variables: $x_{i,j}$, $w_{i,j}$ and $p_{i,j}$, $x_{i,j}$, is a binary number that is equal to 1 if a UAV goes from node i to node j. $w_{i,j}$ is the total weight of the parcels that the UAV is carrying when going from node i to node j. $p_{i,j}$ is the number of parcels that a UAV is carrying when flying from node i to node j . With the presented parameters and the variables, the mathematical formulation of mCVRP is the following:

Objective:

$$min \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j} d_{i,j}(1 + w_{i,j}) \tag{13}$$

Subject to:

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad \forall i = \{2, \ldots, n\} \tag{14}$$

$$\sum_{j=1}^{n} x_{j,i} = 1 \quad \forall i = \{2, \ldots, n\} \tag{15}$$

$$\sum_{j=1}^{n} (w_{j,i} - w_{i,j}) = D_i \quad \forall i = \{2, \ldots, n\} \tag{16}$$

$$\sum_{j=1}^{n} (p_{j,i} - p_{i,j}) = 1 \quad \forall i = \{2, \ldots, n\} \tag{17}$$

$$0 \leq w_{i,j} \leq C_w x_{i,j} \quad \forall i, j = \{1, \ldots, n\} \tag{18}$$

$$0 \leq p_{i,j} \leq C_p x_{i,j} \quad \forall i, j = \{1, \ldots, n\} \tag{19}$$

$$x_{i,i} = 0 \quad \forall i = \{1, \ldots, n\} \tag{20}$$

$$x_{i,j} \in \{0,1\} \quad \forall i, j = \{1, \ldots, n\} \tag{21}$$

The objective function (13) aims to minimize the total travel energy cost of all UAVs. Equation (14) and (15) ensure that only one UAV enters or leaves the node i except for the depot. Equation (16) constrains the variable $w_{i,j}$ according to the customer demand D_i. Equation (17) limits the variable $p_{j,i}$ since each customer only demands one package. Equation (18) and (19) are the key constraints that limit the maximum weight and the number of payloads a UAV can carry at each route. Equation (20) makes sure no nodes will be skipped. Equation (21) limits the forces $x_{i,j}$ become a Boolean value.

B. Adopting Genetic Algorithm

We now adopt genetic algorithm to solve the proposed non-linear programming problem. We would state the approach encoding a set of solutions into chromosome and the design of fitness function in order to implement genetic algorithm in this section. The solution to an mCVRP is sets as: $\{\{0 \to 2 \to 3 \to 5 \to 0\}, \{0 \to 7 \to 6 \to 0\}, \{0 \to 1 \to 4 \to 0\}\}$ As shown in Fig. 4. Each element in the sets indicates the ID of the customer and ID 0 is the depot. We can directly encode the sequence of the solution set by eliminating the depots as Fig. 5. Repeat pattern is not allowed since each customer only receives one parcel in each mission. Each segment of the chromosome is determined by the summation of demands, once $\sum D_i$ is greater than the UAV payload limit C_w, another UAV is needed to deliver the extra parcel. The length of the chromosome is a constant equal to $n-1$. It is worth noticing that different combination with identical fitness values is possible. For example, Fig. 5 shows 5 redundant results having the exact routing result.

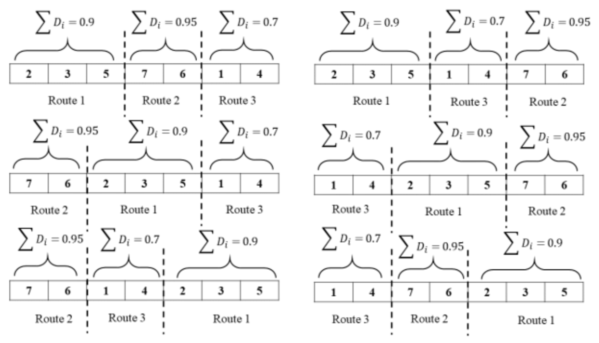


Fig. 5. Example of chromosome encoding.

We establish a function to evaluate the fitness of a chromosome, which is set to be the reciprocal of total energy. The energy could be calculated via (2). We implement a 3-way tournament selection as our selection strategy, selecting 3 individuals randomly and running a tournament among them. Only the fittest candidate is chosen and passed to perform crossover and mutation processes. The details of crossover and mutation operations are described below. The elitism strategy is also adopted to ensure that the fittest solution is always retained without undergoing crossover or mutation operations.

1) Crossover Operation: We now adopt Order Crossover genetic algorithm to solve the proposed non-linear programming problem. To implement Order Crossover (OX), we need 2 chromosomes, denoted as parent 1 and parent 2, in advance. The steps are as follows:

Step (1) Select a random segment from parent 1.

Step (2) Place selected segment at the corresponding position of a new offspring.

Step (3) Remove the element existing in the selected segment from parent 2.

Step (4) Place the rest of part of parent 2 at the unfixed position of offspring 1 from left to right according to the order of the sequence.

Step (5) Go back to Step (1) and swap parents to generate another offspring.

2) Mutation Operation: The mutation operation is performed by arbitrarily selecting a segment of random length in a chromosome and flipping the order of the selected segment. This kind of mutation operation will not break the integrity of the chromosome.

VI. TEST RESULTS

A. DRL-based Method Training Results

The adopted method is trained in a square map with a width of 50 meters and a total of 5 agents. The maximum velocity V_{max} of each agent is 10m/s and the maximum acceleration is 5m/s^2. The sensing range d_{sense} the collider radius d_{colli} of each UAV is set to 15m and 1m respectively. The agent interacts with the environment 50 times per second, i.e., the agent takes 50 actions in a second. The parameters of the SAC method are shown in Table I.

TABLE I. SOFT ACTOR CRITIC HYPERPARAMETERS

Parameters	***Methods/Values***
Optimizer	Adam (Kingma, Jimmy Ba (2015))
Learning Rate	3 \times 10^(-4)
Reward Discount	0.99
Replay Buffer Size	10^6
Number of hidden layers	2
Number of hidden units per layer	256
Number of samples per minibatch	256
Target Entropy	-2
Activation Function	ReLU
Target Smoothing Coefficient	0.005
Temperature Coefficient	0.5

In Fig. 6, we illustrate the learning curve with respect to the episode score of the adopted SAC method. Since the goal of the agent is randomly generated and the total reward is positively correlated with the initial target distance, the score oscillates even if the agent successfully reaches the target. The huge spikes in the unsmoothed curve result from agents failing to avoid collisions and getting tangled with others.

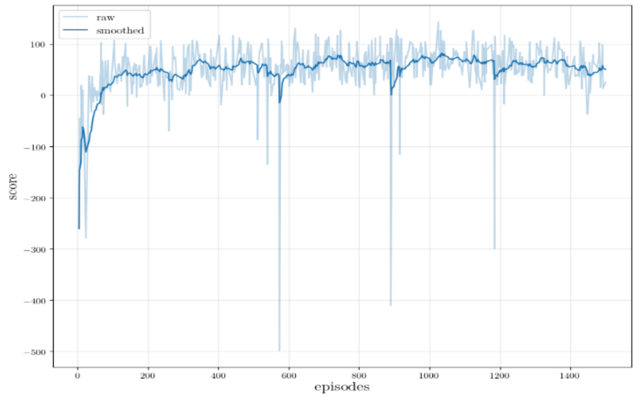


Fig. 6. Training curve of the adopted reinforcement learning algorithm.

In Figs. 7 and 8, we compare the training curves of different deep reinforcement learning methods with respect to the success rate (SR) and the collision rate (CR). SR is defined as the number of agents that successfully reaches their target divided by the number of total agents in the simulation. CR is defined as the number of agents collides with other agents divided by the number of total agents in the environment. The result is produced by running 10 episodes of evaluation every 10 episodes while training. The training result shows that the success rate of the adopted SAC method outperforms other famous deep reinforcement methods such as DDPG and TD3. Not only does the adopted method converge faster, but it also performs relatively stable compared to others. Although DDPG seems to learn faster at the beginning, it fails to learn the collision avoidance policy, causing a high collision rate. As for TD3, the result shows that it may be able to learn a good policy concerning the collision rate, but the success rate still does not converge after 1500 episodes of training. In contrast, SAC converges faster and tends to have a more stable performance after 1000 episodes of training. Compared to the Q-learning method proposed in [11], the SAC method provides not only a higher success rate but also continuous UAV motion control instead of discrete UAV heading control. The overall success rate of the validation result is also higher than the Q-learning method with the same number of agents.

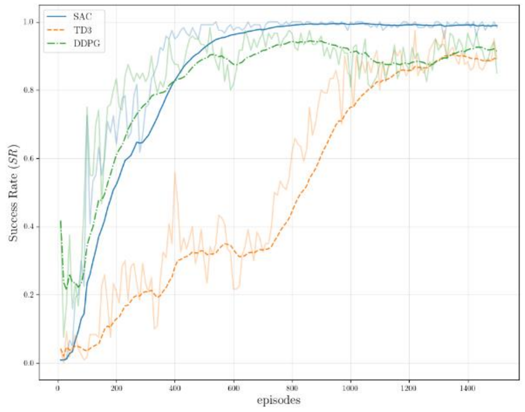


Fig. 7. Success rate with respect to trained episodes of different deep reinforcement learning algorithms including SAC (blue), DDPG (green) and TD3 (orange).

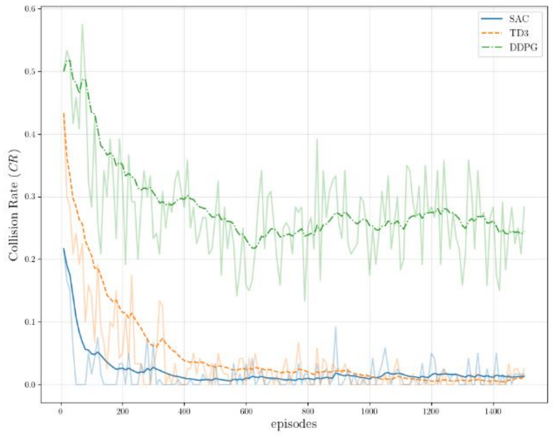


Fig. 8. Collision rate with respect to trained episodes of different deep reinforcement learning algorithms including SAC (blue), DDPG (green) and TD3 (orange).

To further examine the performance of the trained SAC method, we tested the learned policy under environments with different numbers of total agents. Each number of total agents is tested with 100 random episodes. In Fig. 9, it shows that as the total agent number increases, the success rate starts descending, and the collision rate starts ascending. As the environment gets more crowded, the hard limitation of the input state that only allows the agent to perceive at most two nearby obstacles at once becomes more significant. Note that the maximum step of each validation episode is 1500 steps. It is considered failed if an agent is unable to reach its goal within the step limit. The result in Fig. 9 also illustrates the extensibility of the adopted SAC method. The trained policy still has a collision rate below 1 percent and a success rate above 99.9 percent when the total agent number is 12, although the model is trained in an environment with only 5 agents.

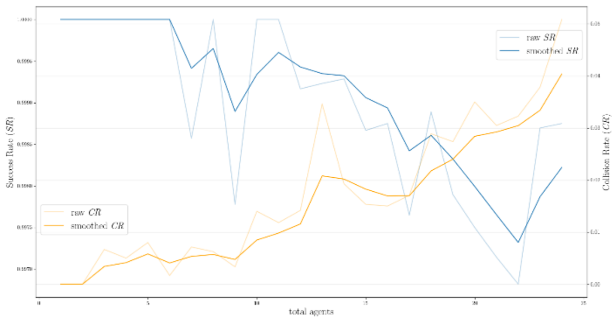


Fig. 9. Success rate(blue) and collision rate(orange) with respect to different numbers of agents in the environment. Light color indicates raw data, while dark color indicates smoothed data.

Finally, we illustrate the result of implementing the proposed DRL-based UAV collision avoidance method with the trained model in a realistic simulation environment using ROS and Gazebo. The maximum velocity V_{max} of each UAV agent is 10m/s and the maximum acceleration is5m/s^2. The sensing range d_sense of the UAVs is set to 15m. The trajectory of four UAV agents trying to avoid collisions between each other using the proposed method while swapping their positions is shown in Fig. 10. The result shows that even if the proposed deep reinforcement learning based model is trained in a simplified simulated environment, it can still be adopted in a realistic physics engine with sensor noises and control delays.

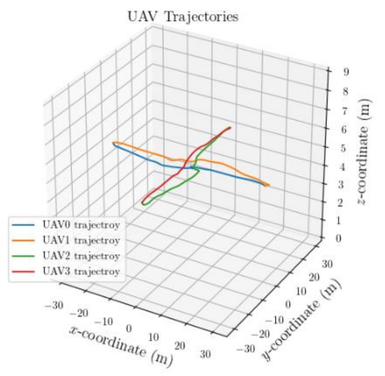


Fig. 10. Trajectory of 4 UAVs trying to avoid each other.

B. Test Results of the Genetic Algorithm Based Route Optimization in Simulated Parcel Delivery System

The genetic algorithm converges after about 1500 generations with the parameters shown in Table II. It took 8.3 seconds to evolve 3000 generations on a laptop with Intel® Core™ i7-12650H CPU. The optimized result leads the UAV to first deliver the heaviest and nearest parcel in order to reduce its overall energy consumption. The routing result of the proposed genetic algorithm in the scenario of 20 customers is shown in Fig. 11. Each blue dot in the figure represents a customer and the number beside it denotes the demand D_i and the center red dot represents the depot.

TABLE. 2 GENETIC ALGORITHM PARAMETERS

Parameters	***Values***
Map Width	1000m
Number of Customers	20
Number of Population	30
Weight Capacity	1.0kh
Maximum number of carried parcels for a UAV	3
Mutation Rate	0.6

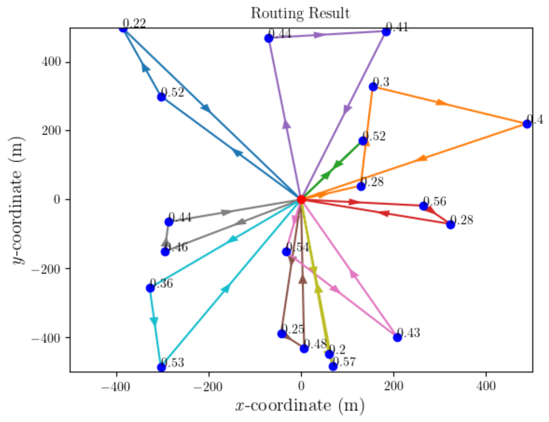


Fig. 11. Demonstration of the proposed genetic algorithm with 20 customers.

The convergence curve of a total of 100 customers with different population sizes is shown in Fig. 12. We can notice that the convergence rate is higher as the population size gets larger; however, the computation time also rises when the population size is increased.

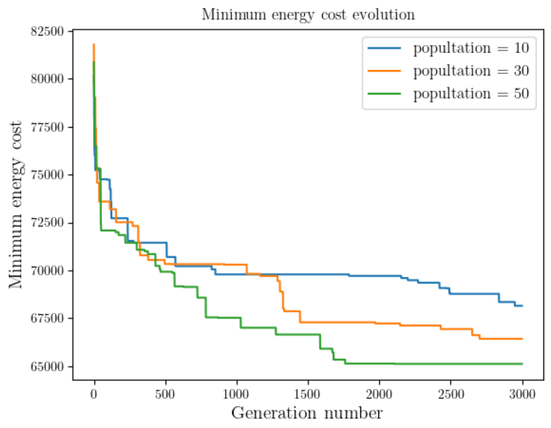


Fig. 12. Convergence curve of 100 customers concerning different population sizes. Blue stands for 10, orange for 30, and green for 50.

To evaluate the performance of the proposed genetic algorithm method, we compared the total energy cost produced by the genetic algorithm with the result using first come first serve scheduling (FCFS). Fig. 13 shows the improvement percentage when utilizing the proposed genetic algorithm with respect to FCFS in 100 missions. Each mission contains 20 customers. The experiment results show that the average improvement percentage is 29.41%.

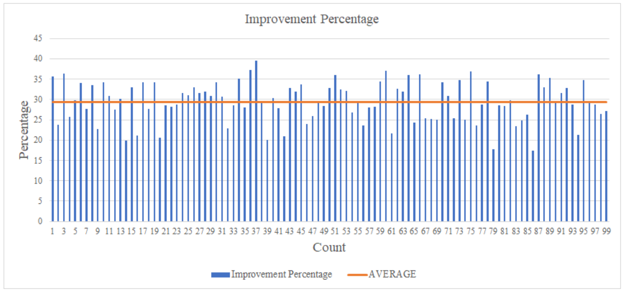


Fig. 13. Improvement our genetic algorithm in comparison to traditional FCFS method. We recorded a 29.41% improvement on average with respect to FCFS method.

C. Adopting the Proposed Methods in a Simulated Parcel Delivery System

We now carry out simulation of the parcel delivery system. We first create a 3-D model of the campus of National Taiwan University (NTU) for the simulation, the model is shown in Fig. 14.

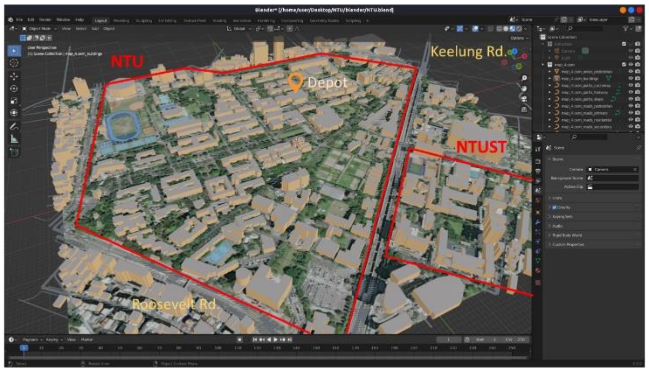


Fig. 14. 3-D model of the NTU campus in Blender

In the simulation, 20 orders with different parcel weight and destination are generated. Then, 10 UAVs will depart the central depot after being loaded, follow the route generated by our genetic algorithm to deliver the parcel to the right position while avoiding collision with each other, and finally go back to the original depot. Each UAV in the simulation can carry 1 kg of payload and 3 parcels at the maximum, flying at a height of h = 100 meters. The scene of UAVs taking off from the depot is shown in Fig. 15, and the route generated by our algorithm is shown in Fig. 16. In the end, it took 5 minutes and 8 seconds for 10 UAVs to deliver 20 parcels to customers at different locations. We now showed the feasibility of applying the proposed UAV collision avoidance method and the genetic algorithm based on route optimization in parcel delivery tasks with the result gained from the simulation.

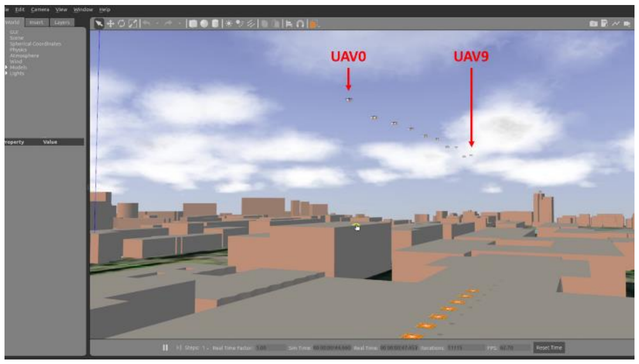


Fig. 15. Illustration of the UAVs taking off and start delivering.

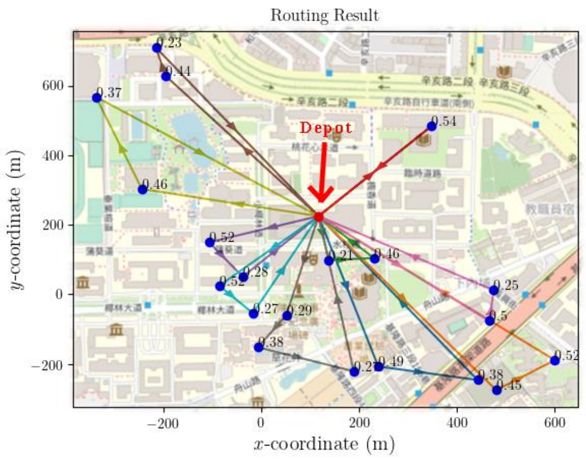


Fig. 16. Routing result of the simulation

VI. CONCLUSION AND FUTURE WORKS

In recent years, more logistics services are making use of the UAV parcel delivery system. However, implementing such a system involves solving some problems such as UAV collision avoidance and route optimization. Therefore, in this thesis, we proposed a UAV collision avoidance solution by adopting a state-of-the-art deep reinforcement learning model. Moreover, the proposed method works in a decentralized manner, it does not require successive communication with the centralized server, and it takes actions according to the local observation of an agent. In addition, to further reduce the cost of the logistic, we proposed a genetic algorithm to generate optimized delivery routes. Instead of minimizing the total distance, we aim to minimize the total energy cost in UAV delivery tasks. We modified the conventional CVRP with additional constraint and solve the optimization problem by using genetic algorithm along with our custom fitness function to obtain optimized routes in UAV delivery tasks. Finally, we validate the proposed method using physics engines and SITL to evaluate the feasibility of our proposed method. The result shows that our proposed UAV collision avoidance method is able to work in a realistic simulation environment. In future research, we aim to validate our UAV collision method on a UAV in a real-world environment. Also, we would like to compare the computational efficiency of the proposed genetic algorithm with some other modern non-linear programming solvers.

REFERENCES

[1] S. R. R. Singireddy and T. U. Daim, ”Technology roadmap: Drone delivery–amazon prime air,” in *Infrastructure and technology manage- ment*, Springer, 2018, pp. 387-412.

[2] I.Hong,M.Kuby,andA.T.Murray,”Arange-restrictedrechargingsta- tion coverage model for drone delivery service planning,” *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 198-212, 2018.

[3] S. M. Shavarani, M. G. Nejad, F. Rismanchian, and G. Izbirak, ”Ap- plication of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San

Francisco," *The International Journal of Advanced Manufacturing Technology*, vol. 95, no. 9, pp. 3141-3153, 2018.

[4] H.HuangandA.V.Savkin,"Deploymentofchargingstationsfordrone delivery assisted by public transportation vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[5] D. Wang, P. Hu, J. Du, P. Zhou, T. Deng, and M. Hu, "Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10483-10495, 2019.

[6] D. N. Das, R. Sewani, J. Wang, and M. K. Tiwari, "Synchronized truck and drone routing in package delivery logistics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5772-5782, 2020.

[7] K.-W. Chen, M.-R. Xie, Y.-M. Chen, T.-T. Chu, and Y.-B. Lin, "DroneTalk: An Internet-of-Things-Based Drone System for Last-Mile Drone Delivery," *IEEE Transactions on Intelligent Transportation Sys- tems*, 2022.

[8] M. Quigley et al., "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, no. 3.2: Kobe, Japan, p. 5.

[9] K.Loayza,P.Lucas,andE.Pelaez,"Acentralizedcontrolofmovements using a collision avoidance algorithm for a swarm of autonomous agents," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017: IEEE, pp. 1-6.

[10] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*, 2012: IEEE, pp. 477-483.

[11] Y.-H. Hsu and R.-H. Gau, "Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 306-320, 2020.

[12] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3098- 3105, 2020.

[13] X. Wang and M. C. Gursoy, "Learning-based UAV trajectory opti- mization with collision avoidance and connectivity constraints," *IEEE Transactions on Wireless Communications*, 2021.

[14] D.Silver,G.Lever,N.Heess,T.Degris,D.Wierstra,andM.Riedmiller, "Determi nistic policy gradient algorithms," in *International conference on machine learning*, 2014: PMLR, pp. 387-395.

[15] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, 2017: IEEE, pp. 316-321.

[16] Y. Hou, J. Zhao, R. Zhang, X. Cheng and L. Yang, "UAV Swarm Cooperative Target Search: A Multi-Agent Reinforcement Learning Approach," in *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 568-578, Jan. 2024.

[17] Y. Xue and W. Chen, "Multi-Agent Deep Reinforcement Learning for UAVs Navigation in Unknown Complex Environment," in *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 2290-2303, Jan. 2024.

[18] L. Xu, T. Wang, J. Wang, J. Liu and C. Sun, "Attention-Based Policy Distillation for UAV Simultaneous Target Tracking and Obstacle Avoid- ance," in *IEEE Transactions on Intelligent Vehicles*

[19] Y. Hui, X. Zhang, H. Shen, H. Lu and B. Tian, "DPPM: Decentralized Exploration Planning for Multi-UAV Systems Using Lightweight Infor- mation Structure," in *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 613-625, Jan. 2024

[20] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86-109, 2015.

[21] Z. Wang and J.-B. Sheu, "Vehicle routing problem with drones," *Transportation research part B: methodological*, vol. 122, pp. 350-364, 2019.

[22] W. Yao et al., "Evolutionary Utility Prediction Matrix-Based Mission Planning for Unmanned Aerial Vehicles in Complex Urban Environ- ments," in *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1068-1080, Feb. 2023.

[23] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70-85, 2016.

[24] N. A. Kyriakakis, T. Stamadianos, M. Marinaki, and Y. Marinakis, "The electric vehicle routing problem with drones: An energy minimization approach for aerial deliveries," *Cleaner Logistics and Supply Chain*, vol. 4, p. 100041, 2022.

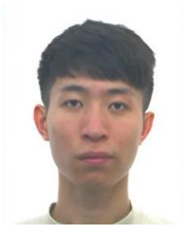
[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off- policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, 2018: PMLR, pp. 1861-1870.

[26] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv*:1509.02971, 2015.

[27] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approxi- mation error in actor-critic methods," in *International conference on machine learning*, 2018: PMLR, pp. 1587-1596.

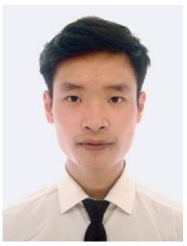
[28] T. Haarnoja et al., "Soft actor-critic algorithms and applications," *arXiv preprint arXiv*:1812.05905, 2018.

[29] G. Brockman et al., "OpenaAI Gym," *arXiv preprint arXiv*:1606.01540, 2016.



Chun-Yuan Chi received M.S. degree in electrical engineering from National Taiwan University of Science and Technology in 2022. His research interests include unmanned aerial vehicles, vehicle path planning and machine learning.

De-Fu Chen is current an undergraduate student of the mechanical engineering, National Taiwan University. His research interests include computer vision, intelligent vehicles and vehicle navigation.



Hoang-Phuong Doan (Nikolas Doan) is a Research Assistant and Master's Student in Control System Division, Department of Mechanical Engineering, National Taiwan University. His research interests include Smart Manufacturing, Additive Manufacturing, and Artificial Intelligence in Robotics.

Chung-Hsien Kuo received the Ph.D. degree in mechanical engineering from National Taiwan University, Taipei, Taiwan, in 1999. He is currently a Professor with the Department of Mechanical Engineering, National Taiwan University, Taipei, Taiwan. His main research areas include autonomous systems, novel soft robot design, sensors design, signal processing, computer vision and machine learning.