2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Novel Variable Step-size Path Planning Framework with Step-Consistent Markov Decision Process For Large-Scale UAV Swarm

Dan Xu[1] , Yunxiao Guo[2], and Han Long[2], Chang Wang[3]

***Abstract*— In recent years, Deep Reinforcement Learning (DRL) has been a key approach to solving Unmanned Aerial Vehicle (UAV) swarm path planning problems. However, traditional DRL methods often face challenges in the initial learning stage and struggle to learn from variable step-size tasks. This paper introduces a novel training framework for large-scale UAV swarm variable step-size path planning: Rapidly-exploring Variable Step-size Deep Reinforcement Learning (RVSDRL). This framework involves common training on the ground local server and decentralized training on distributed UAVs. In the common training stage, we generate rapidly-exploring random graph samples to accelerate the common agent explore environment. In the decentralized training stages, we utilize the priority replay mechanism to improve efficiency. To enhance convergence stability, we restrict the returns of the equivalent paths and propose the Step-size Consistent Markov Decision Process (SCMDP) path planning model. Our method is compared with traditional methods, and the experiments demonstrate its superior performance in complex obstacle environments.**

I. Introduction

Path planning is a classical topic in operations research and optimization, which is widely applied in UAV technology [1]–[3]. Reinforcement Learning has shown its supervising performance in path planning; the model-free DRL has demonstrated advantages in adapting to unknown environments and can be applied in many on-time scenarios [4], [5]. However, the current DRL-based methods often spend much time stochastically exploring the feasible paths and avoiding obstacles at the initial stage [6]. When the number of obstacles is large, and the environment is complex enough, the exploration time at the initial stage will increase rapidly [7]. Meanwhile, the DRL-based methods depend highly on the reward function, and the reward design in complex obstacle environments is an undetermined problem [1]. Besides, the DRL, especially policy gradient methods, also face several challenges, such as lousy convergence performance [5], low training efficiency [2], and low sample utilization efficiency [8].

Recently, researchers have explored novel DRL algorithms and frameworks to address these challenges. In the exploration efficiency aspect, Xie et al. [7] combined the current reward and the state-action value for action selecting, which reduces the meaningless exploration. In the reward design aspect, Zhao et al. [1] designed a complete reward function, which considered the target point, obstacles, and direction simultaneously and tested it at the various obstacle environments. To accelerate the training process, Liao et al. [3] combined the potential field method with the DRL algorithm to deal with the complex input space, decreasing the training processing time. To improve the convergence performance, Dong et al. [2] used prior knowledge to enhance the learning performance, which decreased the convergence time. To improve the sample efficiency, Chen et al. [9] employed the prioritized experience replay technique to change the sample weight, enhancing the DRL algorithm's sampling efficiency.

We propose a novel path-planning framework named Rapidly-exploring Variable Step-size Deep Reinforcement Learning (RVSDRL) and the Step-size Consistent Markov Decision Process (SCMDP) to deal with these issues. The main contributions are as follows:

1) The RVSDRL introduces the rapidly random-exploring graph samples to associate the DRL generate valuable samples and prioritized replay to improve the sample efficiency.
2) The SCMDP restricts the relationship between path and cumulative reward. We also design a path-dependent reward to construct an SCMDP in the path-planning scenario and analyze their convergence.
3) RVSDRL outperforms DDPG-based methods in selecting step size, planning the shortest path, and performing better in dense obstacle environments, especially at the initial stage.

II. Preliminaries

A. Problem Formulation

The UAV swarm path planning problem can be considered on a configuration space $\chi := (0,1)^d$, where the dimension $d \in \mathbb{N}, d \geq 1$. Denote χ_{obs} as the obstacle region, denoted the obstacle-free space as $\chi_{\text{free}} = \text{cl}(\chi \setminus \chi_{\text{obs}})$, where $\text{cl}(\cdot)$ represents the closure space. The initial region χ_{init} and the goal region χ_{goal} are both elements of χ_{free}, which represent the UAVs' start points and endpoints. Then, a path planning problem can be defined as a triplet $(\chi_{\text{free}}, \chi_{\text{init}}, \chi_{\text{goal}})$. The feasible path is the path that connects the initial point and goal point, and the path planning target is to find a feasible path or an optimal path in the given problem $(\chi_{\text{free}}, \chi_{\text{init}}, \chi_{\text{goal}})$.

[1]Dan Xu is with the School of System Science and Engineering, Sun-Yat-Sen University, Guangzhou, 510006, Guangdong, China, and College of System Engineering, National University of Defense Technology, 410073 Changsha, China xudan_changsha@163.com

[2]Yunxiao Guo and Han Long are with the College of Science, National University of Defense Technology, Changsha, China, 410073 guoyunxiao.nudt@hotmail.com, longhan@nudt.edu.cn

[3]Chang Wang is with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China, 410073 wangchang07@nudt.edu.cn

Chang Wang is the corresponding author.

B. Offline Reinforcement Learning

A Markov Decision Process (MDP) [10] $\mathcal{M}$ can be descripted as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is state space, $\mathcal{A}$ is action space, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. The agent interacts with the task environment, generates the state, action, and reward information, and collects them. Then, the MDP maximizes the following discounted return on each time step t to reach the global optimal solution:

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots = \sum_{k=0}^{+\infty} \gamma^k r_{t+k}. \quad (1)$$

To reach the target, the offline Reinforcement Learning uses the well-collected data buffer $\mathcal{D} = \{(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')\}$ to optimize the policy of the agent [10]. Denote the policy as π, the corresponding value function can be represented as:

$$V_\pi^{\mathcal{D}}(\boldsymbol{s}_t) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\sim\mathcal{D}}\left[\sum_k^{+\infty} \gamma^k r_{t+k} | \boldsymbol{s} = \boldsymbol{s}_t\right], \quad (2)$$

$$Q_\pi^{\mathcal{D}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\sim\mathcal{D}}\left[\sum_k^{+\infty} \gamma^k r_{t+k} | \boldsymbol{s} = \boldsymbol{s}_t, \boldsymbol{a} = \boldsymbol{a}_t\right]. \quad (3)$$

Then, the agent utilizes the value function to associate the policy's improvement with maximizing their return.

C. Variable Step-size Planning Path with Reinforcement Learning

For any path planning problem $(\chi_{\text{free}}, \chi_{\text{init}}, \chi_{\text{goal}})$, denote the planning space $\chi \subset \mathcal{S}$ as the state space, denote the current position $\sigma(t_k)$ as current state $\boldsymbol{s}_k$, then denote the transition between current and next states as action $\boldsymbol{a}_k$: $\boldsymbol{s}_k \to \boldsymbol{s}_{k+1}$. Denote the $\Delta \boldsymbol{s}_k$ as the step-size at kth time point, $\Delta \boldsymbol{s}_k \in \mathbb{R}^1, \Delta \boldsymbol{s}_k \in [\Delta \boldsymbol{s}_{\min}, \Delta \boldsymbol{s}_{\max}]$; denote $\vec{d}_k \in \mathbb{R}^d$ as the unit direct vector at kth time step, $\|\vec{d}_k\| = 1$. Then the action $\boldsymbol{a}_k \in \mathbb{R}^{d+1}$ can be represented as the set of step-size and direct vector:

$$\boldsymbol{a}_k := \begin{bmatrix} \Delta \boldsymbol{s}_k \\ \vec{d}_k \end{bmatrix}, \quad (4)$$

Therefore, the state transition can be represented as follows:

$$\boldsymbol{s}_{k+1} = \boldsymbol{s}_k + \vec{d}_k \cdot \Delta \boldsymbol{s}_k. \quad (5)$$

Due to the Markov property:

$$\mathcal{P}\{\boldsymbol{s}_{k+1} | \boldsymbol{s}_k, \boldsymbol{s}_{k-1}, \cdots\} = \mathcal{P}\{\boldsymbol{s}_{k+1} | \boldsymbol{s}_k\}, \quad (6)$$

as we define the reward function r, the variable step-size path planning problem can be described as an MDP. Then, the path planning problem can be optimized using reinforcement learning algorithms.

III. Our Approach

A. Rapidly-exploring Variable Step-size Deep Reinforcement Learning

With the progress of satellite technology such as Starlink, satellites can provide geographic information to mobile devices. Inspired by it, we design a scenario (see Fig. 1) consisting of satellites, a UAV swarm, and a local server. The local server receives the geographic information around the UAV swarm and then sends the information on obstacles and goal regions to associate the UAV with planning the path. By using satellites, the χ_{obs} and χ_{goal} are determined. After that, according to the traditional DRL-based swarm path planning methods, a virtual environment is constructed before the execution of the UAV swarm. Then, the UAV swarm will train in this virtual environment until convergence.

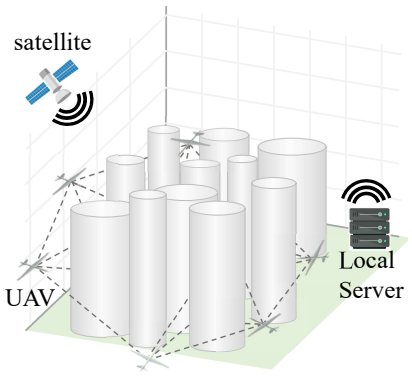

Fig. 1. The diagram of the large-scale UAV swarm path planning

In the training process, the traditional methods take decentralized stochastic exploration to find feasible paths, which makes it easy to hit obstacles and hard to approach the goal region. This stochastic exploration increases the computational consumed in the initial learning stage, especially for the large-scale UAV swarm. To deal with the challenge, we design a novel DRL framework named Rapidly-exploring Variable Step-size Deep Reinforcement Learning (RVSDRL), which generates the common samples on the local server and distributed training on the UAVs. The pseudocode is shown in Algorithm 1.

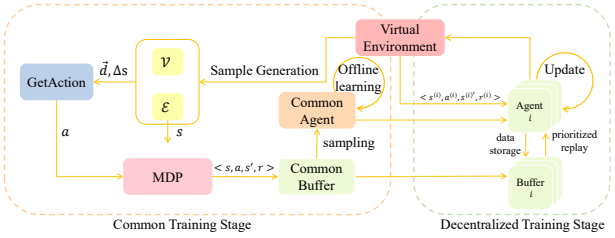

Fig. 2. The workflow of the RVSDRL framework

As shown in Fig. 2, the framework can be divided into common and decentralized training stages. The common training stage is processed on the local server, and the

decentralized training stage is processed independently on each UAV. The details of these two stages are shown as follows:

Common Training Stage: In the common training stage, we introduce the Rapidly Exploring Random Graph to reduce the initial exploration. Specifically, in the virtual environment, given an initialized vertices set: $\mathcal{V} = \{\chi_{\text{goal}}\}$ and an initialized edge set $\mathcal{E} = \varnothing$, then generate the node sample $\boldsymbol{s}_{\text{rand}}$ and a step-sizes Δs_{rand} stochastically and append the generated vertices into $\mathcal{V}$. Next, the nearest node of $\mathcal{V}$ will be selected, and we denote it as $\boldsymbol{s}_{\text{nearest}}$. At this time, connecting the two nodes: $\boldsymbol{s}_{\text{nearest}}, \boldsymbol{s}_{\text{rand}}$, we can obtain a new node $\boldsymbol{s}'$ by take $\Delta \boldsymbol{s}_{\text{rand}}$ step on the line. After the samples are generated, we transfer the collected graph into the experience replay buffer that consists of transitions. Specifically, for each edge e in $\mathcal{E}$, their two vertices will be extracted as $\boldsymbol{s}, \boldsymbol{s}'$, respectively. Then, according to the step-size range, calculate the action $\boldsymbol{a} := \{\Delta \boldsymbol{s}, \vec{d}\}$ as follows:

$$\Delta \boldsymbol{s} = \text{clip}(\|\boldsymbol{s}' - \boldsymbol{s}\|, \Delta \boldsymbol{s}_{\min}, \Delta \boldsymbol{s}_{\max}) \tag{7}$$

$$\vec{d} = \frac{(\boldsymbol{s}' - \boldsymbol{s})}{\|\boldsymbol{s}' - \boldsymbol{s}\|} \tag{8}$$

Next, the processed $\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'$ will be checked collision or done and generate the transitions; the generated transitions are collected with a buffer $\mathcal{B}$. Then, the local server will utilize the collected buffer to train a common agent $\text{Agent}_{\text{com}}$. The above processes will be repeated after N_{Inc} times. Because the samples generated by this stage cover all regions, they can provide valuable information to the UAV at any position. Compared with the Rapidly-exploring Random Graph (RRG), this process employs an incremental generation approach, allowing the simultaneous generation of samples and DRL model training when new edges are created. Moreover, this process removes the collision detection in the RRG, increasing the sample generation efficiency.

Decentralized Training Stage: In this stage, each UAV of the swarm receives the common agent parameters and virtual environment settings from the local server and then starts simulative training in a distributed manner. During the simulative training, the newly generated transitions will be collected to each buffer $\mathcal{B}_i$. Finally, the UAV agent will perform offline learning on each buffer.

Besides, we introduce the prioritized replay into the offline learning stage to improve sample utilization efficiency. Specifically, the transition produced in all stages will be stored in the replay buffer with the absolute value of the TD error:

$$\delta = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a}) \tag{9}$$

Then, after the transitions are stored into the buffer $\mathcal{B}$, for the t th transition, their sampled probability can be computed as follows:

$$\mathcal{P}(t) = \frac{D_t^\alpha}{\sum_{k=1}^{\text{card}(\mathcal{B})} D_k^\alpha} \tag{10}$$

where $D_t = \frac{1}{\text{rank}(t)} > 0$, $\text{rank}(t)$ is the rank of the t-th transition in the buffer with absolute TD-error (9) being the criterion, α is the constant to adjust the distribution of the probability.

Algorithm 1 RVSDRL

Input: χ_{goal}, χ_{free}, N, N_{epi}, N_{Inc}, $\Delta \boldsymbol{s}_{\max}$, $\Delta \boldsymbol{s}_{\min}$.
Output: trained $\text{Agent}_1, \cdots, \text{Agent}_N$.
Common Training Stage:
$\mathcal{V} \leftarrow \{\chi_{\text{goal}}\}$; $\mathcal{E} \leftarrow \{\}$; $\mathcal{B} \leftarrow \{\}$.
for $i = 1, 2, \cdots, N_{\text{Inc}}$ **do**
  $\boldsymbol{s}_{\text{rand}} \leftarrow \textbf{Sample}(\chi_{\text{free}})$;
  $\Delta \boldsymbol{s}_{\text{rand}} \leftarrow \textbf{rand}(\Delta \boldsymbol{s}_{\min}, \Delta \boldsymbol{s}_{\max})$
  $\boldsymbol{s}_{\text{nearest}} \leftarrow \textbf{Nearest}(\mathcal{V}, \boldsymbol{s}_{\text{rand}})$
  $\boldsymbol{s}' \leftarrow \textbf{Steer}(\boldsymbol{s}_{\text{nearest}}, \boldsymbol{s}_{\text{rand}}, \Delta \boldsymbol{s}_{rand})$
  $\boldsymbol{S}_{\text{near}} \leftarrow \textbf{Near}(\mathcal{V}, \boldsymbol{s}', \min\{\Delta \boldsymbol{s}_{\max}, \varsigma_{\text{RRG}} \cdot [\ln(\text{card}(\mathcal{V}))/\text{card}(\mathcal{V})]^{1/d}\})$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{\boldsymbol{s}'\}$
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\boldsymbol{s}_{\text{nearest}}, \boldsymbol{s}'), (\boldsymbol{s}', \boldsymbol{s}_{\text{nearest}})\}$
  for $\boldsymbol{s}_{\text{near}} \in S_{\text{near}}$ **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\boldsymbol{s}_{\text{near}}, \boldsymbol{s}'), (\boldsymbol{s}', \boldsymbol{s}_{\text{near}})\}$
  end for
  for $e \in \mathcal{E}$ **do**
    $\boldsymbol{s} \leftarrow e_s$; $\boldsymbol{s}' \leftarrow e_d$;
    $\vec{d}, \Delta \boldsymbol{s} \leftarrow \textbf{GetAction}(\boldsymbol{s}, \boldsymbol{s}', \Delta \boldsymbol{s}_{\min}, \Delta \boldsymbol{s}_{\max})$
    $\boldsymbol{s}, r, \text{coll}, \text{done} \leftarrow \textbf{Step}(\boldsymbol{s}, \vec{d}, \Delta \boldsymbol{s}, \chi_{\text{goal}})$
    $\{\vec{d}, \Delta \boldsymbol{s}\}$
    Update $\boldsymbol{s}'$ according to (5).
    $\text{coll} \leftarrow \textbf{CheckCollision}(\boldsymbol{s}, \boldsymbol{s}')$
    Calculate r according to (12).
    $\text{done} \leftarrow \textbf{CheckDone}(\boldsymbol{s}', \chi_{\text{goal}})$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r)\}$
  end for
  $\text{Agent}_{\text{com}}.\text{train}(\mathcal{B})$
end for
$\text{Agent}_j \leftarrow \text{Agent}_{\text{com}}, j = 1, 2, \cdots, N$
Decentralized Training Stage:
for $i = 1 : N_{\text{epi}}$ **do**
  for $j = 1, 2, \cdots, N$ **do**
    $\mathcal{B}_j^{\text{batch}} \leftarrow \{\}, \mathcal{B}_j \leftarrow \mathcal{B}$
    Agent_j interact with virtual environment and collect the transitions into $\mathcal{B}_j$
    Sample $n_{\mathcal{B}}$ transitions with (10) from $\mathcal{B}_j$
    Add sampled results into $\mathcal{B}_j^{\text{batch}}$
    $\text{Agent}_j.\text{train}(\mathcal{B}_j^{\text{batch}})$
  end for
end for

B. Planning Path with Step-size Consistent Markov Decision Process

The current reward function makes it hard to keep the path return consistent if the two coincident paths' step-sizes are different (see Fig. 3), then their corresponding returns (1) may not be equal: the paths with more steps tend to accumulate more rewards. The difference in the returns will cause the coincident paths to have different Q values and increase the return difference of the similar paths. It increases the complexity of the search space.

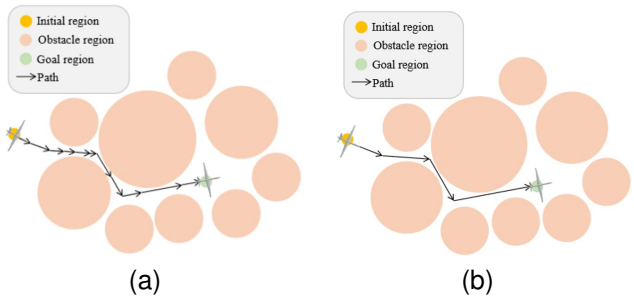

Fig. 3. The coincident paths with different step-sizes

To deal with this issue, we build the relationship between the returns and paths' metric by introducing the Step-size Consistent Markov Decision Process (SCMDP):

Definition 1 (SCMDP): In a variable path planning problem with an MDP $\mathcal{M}$, for any two paths σ_i, σ_j that with same start point $\boldsymbol{s}_{\text{init}}$ and goal state $\boldsymbol{s}_{\text{goal}}$. If there lengths are equal: $\mathrm{L}(\sigma_i) = \mathrm{L}(\sigma_j)$, and exists an $\xi \in \mathbb{R}^+$, such that the following inequality satisfies:

$$\|R(\sigma_i) - R(\sigma_j)\| \leq \xi \cdot \mathrm{D}(\sigma_i, \sigma_j) \tag{11}$$

where $R(\sigma)$ is the return of path σ at the end point, $\mathrm{D}(\cdot,\cdot)$ is the similarity metric between the paths. Then the $\mathcal{M}$ is a Step-size Consistent MDP concerning $\boldsymbol{s}_k$ and $\boldsymbol{s}_{k+n}$.

The SCMDP restricts the paths' corresponding returns when they are close in geometry, reducing the agent's search complexity. Especially for any two entirely coincident paths σ_i, σ_j, even if their step-sizes are different, the corresponding returns are equal: $R(\sigma_i) = R(\sigma_j)$.

C. Path-dependent Reward and Discount Factor

To construct an SCMDP, we propose the path-dependent reward as follows:

$$r_k = \int_{\boldsymbol{s}_k}^{\boldsymbol{s}_{k+1}} \rho(\boldsymbol{s}) \gamma_0^{\|\boldsymbol{s}-\boldsymbol{s}_k\|/\kappa} d\boldsymbol{s}. \tag{12}$$

where $\boldsymbol{s}_k := \sigma(t_k)$ is the state at time step k, $\gamma_0 \in (0,1)$ is the baseline discount factor, $\kappa \in \mathbb{R}^+$ is the scale parameter, $\rho : \chi \to \mathbb{R}^1$ can be any custom Lipschitz continuous distribution function respect to the state $\boldsymbol{s}$, a typical form is consisted of the potential function part and the region reward part, given by:

$$\rho(\boldsymbol{s}) = -C_\rho \frac{\|\boldsymbol{s}_k - \boldsymbol{s}_{\text{goal}}\|}{\|\boldsymbol{s}_0 - \boldsymbol{s}_{\text{goal}}\|} + r_{\text{region}}(\boldsymbol{s}) \tag{13}$$

where $C_\rho \in \mathbb{R}^+$ is a constant, the potential function part is utilized to guide the agent to approach the goal constantly. The region rewards r_{region} is designed to give the agent different stimuli at the different positions, which is represented as follows:

$$r_{\text{region}}(\boldsymbol{s}) = \begin{cases} C_{\text{goal}}, \boldsymbol{s} \in \chi_{\text{goal}} \\ -C_{\text{obs}}, \boldsymbol{s} \in \chi_{\text{obs}} \\ 0, \text{otherwise} \end{cases} \tag{14}$$

where $C_{\text{goal}}, C_{\text{obs}} \in \mathbb{R}^+$ are positive constants. The path-dependent reward function r_k, characterizes the reward progression along a trajectory with exponential attenuation, determined by the spatial span between the previous and current states. Simultaneously, the discount factor γ is redefined in a path-dependent form:

$$\gamma_k = \gamma_0^{\|\Delta \boldsymbol{s}_k\|/\kappa} \tag{15}$$

For the proposed reward and discount factor, the corresponding cumulative return is also path-dependent and can be expressed as follows:

$$R_t := R_t(\sigma) = \sum_{k=t}^{+\infty} \Big(\prod_{v=t}^{k-1} \gamma_v \Big) \cdot r_k \tag{16}$$

when the agent reaches the goal region or obstacle region: $\boldsymbol{s}_k \in \chi_{\text{goal}} \cup \chi_{\text{obs}}$, we terminate the agent.

D. Analysis

Combining the Eq. (12) and (15), the return (16) can be represented as follows:

$$\begin{aligned} R_t &= \sum_{k=t}^{+\infty} \prod_{v=t}^{k-1} \gamma_0^{\|\Delta \boldsymbol{s}_v\|/\kappa} \int_{\boldsymbol{s}_k}^{\boldsymbol{s}_{k+1}} \rho(\boldsymbol{s}) \gamma_0^{\|\boldsymbol{s}-\boldsymbol{s}_k\|/\kappa} d\boldsymbol{s} \\ &= \sum_{k=t}^{+\infty} \int_{\boldsymbol{s}_k}^{\boldsymbol{s}_{k+1}} \rho(\boldsymbol{s}) \gamma_0^{(\sum_{v=t}^{k-1} \|\Delta \boldsymbol{s}_v\| + \|\boldsymbol{s}-\boldsymbol{s}_k\|)/\kappa} d\boldsymbol{s} \end{aligned} \tag{17}$$

for the time step t, denote the piece-wise function $\mathrm{D}_\sigma : \chi \times \chi \to \mathbb{R}^+$ as follows:

$$\begin{aligned} &\mathrm{D}_\sigma(\boldsymbol{s}_t, \boldsymbol{s}) \\ &= \begin{cases} \|\boldsymbol{s} - \boldsymbol{s}_t\|, \boldsymbol{s} \in [\boldsymbol{s}_t, \boldsymbol{s}_{t+1}] \\ \|\boldsymbol{s}_{t+1} - \boldsymbol{s}_t\| + \|\boldsymbol{s} - \boldsymbol{s}_{t+1}\|, \boldsymbol{s} \in [\boldsymbol{s}_{t+1}, \boldsymbol{s}_{t+2}] \\ \vdots \\ \sum_{v=t}^{k-1} \|\boldsymbol{s}_{v+1} - \boldsymbol{s}_v\| + \|\boldsymbol{s} - \boldsymbol{s}_k\|, \boldsymbol{s} \in [\boldsymbol{s}_k, \boldsymbol{s}_{k+1}] \\ \vdots \end{cases} \end{aligned}$$

then the Eq. (17) can be represented as follows:

$$R_t = \sum_{k=t}^{+\infty} \int_{\boldsymbol{s}_k}^{\boldsymbol{s}_{k+1}} \rho(\boldsymbol{s}) \gamma_0^{\mathrm{D}_\sigma(\boldsymbol{s}_t, \boldsymbol{s})/\kappa} d\boldsymbol{s} \tag{18}$$

According to the definition of line-integral, the return can be rewritten as a line-integral formation:

$$R_t = \int_{\sigma_t} \rho(\boldsymbol{s}) \gamma_0^{\mathrm{D}_\sigma(\boldsymbol{s}_t, \boldsymbol{s})/\kappa} d\boldsymbol{s} \tag{19}$$

The line-integral formation illustrates the return is path-dependent. Moreover, the Eq. (19) indicates the physical significance of the path-dependent reward: **integrate the reward density along the path σ_t with exponential decay based on the distance from the starting point $\boldsymbol{s}_t$.**

Next, we give the following theorem to prove that for a continuous space path planning problem, if their return is defined as Eq. (19), the path planning process is an SCMDP (All the proof is shown in supplementary material[1]).

[1] https://www.researchgate.net/publication/379000186_supplementary_material

Theorem 1: In a path planninof g task with MDP $\mathcal{M}$, if its return is defined by Eq. (16), then $\mathcal{M}$ is an SCMDP concerning any other point to $\boldsymbol{s}_{\text{goal}}$ in Euclidean distance.

It indicated that the MDP $\mathcal{M}$ with a designed reward (12) and discount factor (15) is a SCMDP. Next, we discuss the convergence of the SCMDP:

Theorem 2: For a SCMDP $\mathcal{M}$ with variable discount (15), if the baseline discount $\gamma_0 \in (0, 1)$, step-size $\Delta \boldsymbol{s} \in [\Delta \boldsymbol{s}_{\min}, \Delta \boldsymbol{s}_{\max}]$, and $\|\Delta \boldsymbol{s}_{\min}\| > 0$, then $\mathcal{M}$ is convergent:

$$\|\mathcal{B}Q_1 - \mathcal{B}Q_2\|_\infty \leq \gamma_0^{\|\Delta \boldsymbol{s}_{\min}\|/\kappa} \|Q_1 - Q_2\|_\infty \tag{20}$$

where $\mathcal{B}$ is the Bellman operator, $\|\boldsymbol{x}\|_\infty$ is the sup-norm of vector $\boldsymbol{x}$; $\gamma_0^{\|\Delta \boldsymbol{s}_{\min}\|/\kappa} < 1$ is the convergent rate.

It indicated that the SCMDP with the proposed reward function is convergent, and its' convergent rate is related to the initial discounted factor and the minimum step size.

IV. Experiments

A. Experimental Setting

To validate the RVSDRL framework, we selected the Deep Deterministic Policy Gradient (DDPG) algorithm as the DRL part and called the corresponding algorithm RVSDDPG, in which the step-size range is $(0.01, 2)$; the path planning environments consistent with a swarm with 100 UAVs, 11 circle obstacles and 1 goal region. From the perspective of the comparison set, we implemented the DDPG with Fixed step sizes (DDPG-FS) and DDPG with variable step-size (DDPG-VS) as comparisons. To take ablation experiments, RVSDDPG with Fixed Discount (RVSDDPG-FD) and RVS-DDPG without SCMDP (RVSDDPG*) were implemented as comparisons. The properties of all algorithms are shown in Tab. I.

TABLE I
Comparison of the experiment algorithms

Algorithm	SCMDP	Variable Step-size	Variable discount	RVSDRL
RVSDDPG	✓	✓	✓	✓
RVSDDPG-FD		✓		✓
RVSDDPG*		✓	✓	✓
DDPG-VS		✓		
DDPG-FS				

B. Training Results

We took common training with 5,000 episodes and then took decentralized training with 35,000 episodes on the designed scenario. Fig. 4 (a) illustrates the average reward of 100 UAVs during whole training stage. For the fixed step-size methods, the average reward of DDPG-FS-0.6 is lower than the DDPG-FS-2 before 5,000 episodes. But after common training, the DDPG-FS-0.6 surpassed the DDPG-FS-2 and converged gradually. It indicates that the large step size would limit the performance, but the small step size is slow to converge. For the variable step-size methods, the RVSDDPG algorithm shows the best performance among all the algorithms, which rapidly increases from 0 to 60 near. Compared with RVSDDPG, the RVSDDPG-FD algorithm does not satisfy the SCMDP condition because we set their discount factor as a constant. The experiment results show that its stability has decreased compared with RVSDDPG; its average reward fluctuated three times from 10,000 to 20,000 episodes. After 20,000 episodes, there appeared to be a sharp fluctuation, but it recovered quickly. The RVSDDPG* algorithm doesn't satisfy the SCMDP condition but with a variable discount. It performs better than the RVSDDPG-FD, but the learning curve continuously vibrates, which indicates that without SCMDP, the introduction of the graph sample will cause more fluctuations. The DDPG-VS does not satisfy the SCMDP condition and is without the support of the RVSDRL framework compared with RVSDDPG. The experiment result indicates that it can not deal with complex obstacle scenarios, even if it has a good beginning; after training with stochastic exploration, its average reward collapses heavily and does not tend to converge. From the perspective of convergence performance, the DDPG-FS-2 algorithm converges faster than the DDPG-FS-0.6 and the algorithms with variable step sizes. The RVSDDPG is more stable than the algorithms with variable step sizes but without SCMDP, which validates that the SCMDP can improve convergence stability.

To sum up, the RVSDDPG algorithm shows rapid learning ability in the initial stage, and it becomes stable after the average reward rises, which validates the SCMDP and RVSDRL framework, which can improve the performance of the variable step-size method significantly.

C. Testing Results

After training, we tested the RVSDDPG, DDPG-VS, and DDPG-FS algorithms and visualized the results in Fig. 4 (b)-(d), respectively. Compared with the comparisons, the paths planned by RVSDDPG are smoother and shorter. In Fig. 4 (b)-(d), we divided the planning space into a 50×50 grid to visualize the trained agent's comprehension at each position. We used each grid cell's color to represent the value function $V(\boldsymbol{s})$. The blue represents a low value, the yellow represents a high value, the positions in the obstacle area are lower than the other positions, and as the position approaches the goal area, the value is higher.

We display test data of the well-trained agent in Tab. II. The illustration of the indexes is shown as follows:

- **Average Reward:** the 100 agents' average reward during 1,000 episodes.
- **Completeness Rate:** the proportion of the feasible paths in each episode.
- **Path Difference:** the length of the algorithm planned path subtracts the corresponding of the well-trained RRT* planned.

After training, the RVSDDPG algorithm achieved the best performance among all the algorithms with 56.206 average rewards. The RVSDDPG-FD and RVSDDPG* achieved 48.885 and 45.834, respectively. It indicates that without the SCMDP, the RVSDRL can not reduce the exploration time well in the initial stage. The DDPG-VS is 25.161, which is lower than other algorithms. It reflect the variable step-size

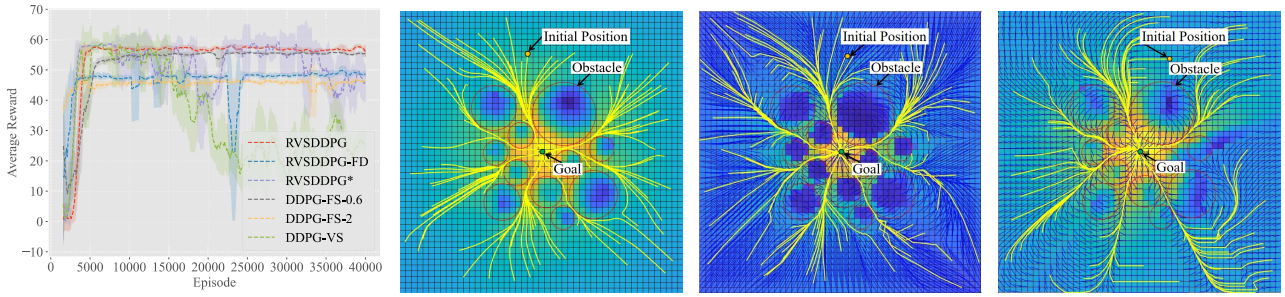

Fig. 4. The learning curves and the completeness rate curves

swarm path planning is very difficult for the traditional DRL method. The DDPG-FS-0.6 is 55.317, and the DDPG-FS-2 is 46.375. It indicates that the performance of the fixed step-size method is very dependent on the selection of the step-size.

From the completeness rate perspective, only RVSDDPG and DDPG-FS achieve 90% above after training, which indicates that the RVSDRL framework can utilize the advantage of the RRG samples to associate the agent in planning the feasible paths. Besides, the RVSDDPG-FD only reached 26.8%, which reflects that the path-dependent reward can not guide the agent to explore feasible paths when the discount is fixed, and the fixed discount will break the SCMDP property, which leads to a decrease in stability.

The path difference index would be combined with the completeness rate to illustrate the length of the feasible paths planned by the agents. For example, the RVSDDPG-FD reached the -0.201 after training. However, the completeness rate of RVSDDPG-FD is only 26.8%, which reveals the short paths in RVSDDPG-FD are not feasible. Combining with the completeness rate and path difference, the RVSDDPG shows its can not only guide the UAVs find the feasible paths with high confidence, but also plan the paths that shorter than RRT*.

TABLE II

THE TEST RESULTS OF PROPOSED ALGORITHM AND COMPARISONS

Algorithm	Average Reward	Completeness Rate	Path Difference
RVSDDPG	$\mathbf{56.206 \pm 1.929}$	$\mathbf{0.939 \pm 0.073}$	$\mathbf{-0.305 \pm 0.08}$
RVSDDPG-FD	48.885 ± 3.326	0.268 ± 0.051	-0.201 ± 0.194
RVSDDPG$*$	45.834 ± 5.959	0.579 ± 0.184	0.056 ± 0.167
DDPG-VS	25.161 ± 4.172	0.313 ± 0.117	0.351 ± 0.316
DDPG-FS-0.6	55.317 ± 1.231	0.912 ± 0.067	-0.039 ± 0.123
DDPG-FS-2	46.375 ± 0.584	0.646 ± 0.137	0.401 ± 0.196

V. CONCLUSION

This paper researched the variable step-size path planning problem. To improve the exploration efficiency, this paper proposed the RVSDRL framework, which utilizes RRG samples for common agent training. Moreover, we built a strong relationship between paths and returns to deal with the inconsistent returns and proposed the SCMDP path planning model. The experiment results show that the RVSDRL framework can be associated with the exploration, and the SCMDP can improve stability compared with traditional methods.

REFERENCES

[1] Y. Zhao, X. Wang, R. Wang, Y. Yang, and F. Lv, "Path planning for mobile robots based on tpr-ddpg," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[2] Y. Dong and X. Zou, "Mobile robot path planning based on improved ddpg reinforcement learning algorithm," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020, pp. 52–56.

[3] X. Liao, Y. Wang, Y. Xuan, and D. Wu, "Agv path planning model based on reinforcement learning," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 6722–6726.

[4] M. A. Anwar and A. Raychowdhury, "Navren-rl: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images," in *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2018, pp. 1–6.

[5] Z. Chu, F. Wang, T. Lei, and C. Luo, "Path planning based on deep reinforcement learning for autonomous underwater vehicles under ocean current disturbance," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 108–120, 2023.

[6] P. Morere, G. Francis, T. Blau, and F. Ramos, "Reinforcement learning with probabilistically complete exploration," *CoRR*, vol. abs/2001.06940, 2020. [Online]. Available: https://arxiv.org/abs/2001.06940

[7] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24 884–24 900, 2021.

[8] S. Y. Luis, F. Peralta Samaniego, D. G. Reina, and S. Toral Marín, "A sample-efficiency comparison between evolutionary algorithms and deep reinforcement learning for path planning in an environmental patrolling mission," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 71–78.

[9] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," *Neurocomputing*, vol. 497, pp. 64–75, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.05.006

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 2018.