Subtask-masked curriculum learning for reinforcement learning with application to UAV maneuver decision-making

Hou, Yueqi; Liang, Xiaolong; Lv, Maolong; Yang, Qisong; Li, Yang

DOI
10.1016/j.engappai.2023.106703

Publication date
2023

Document Version
Final published version

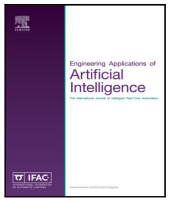
Published in
Engineering Applications of Artificial Intelligence

Engineering Applications of Artificial Intelligence 125 (2023) 106703

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Subtask-masked curriculum learning for reinforcement learning with application to UAV maneuver decision-making

Yueqi Hou [a,b], Xiaolong Liang [a,b,*], Maolong Lv [a], Qisong Yang [c], Yang Li [c]

[a] Air Traffic Control and Navigation School, Air Force Engineering University, Xi'an, China
[b] Shaanxi Key Laboratory of Meta-Synthesis for Electronic and Information System, Air Force Engineering University, Xi'an, China
[c] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands

ARTICLE INFO

Keywords:
Unmanned Aerial Vehicle
Maneuver decision-making
Reinforcement learning
Curriculum learning
Knowledge transfer

ABSTRACT

Unmanned Aerial Vehicle (UAV) maneuver strategy learning remains a challenge when using Reinforcement Learning (RL) in this sparse reward task. In this paper, we propose Subtask-Masked curriculum learning for RL (SubMas-RL), an efficient RL paradigm that implements curriculum learning and knowledge transfer for UAV maneuver scenarios involving multiple missiles. First, this study introduces a novel concept known as subtask mask to create source tasks from a target task by masking partial subtasks. Then, a subtask-masked curriculum generation method is proposed to generate a sequenced curriculum by alternately conducting task generation and task sequencing. To establish efficient knowledge transfer and avoid negative transfer, this paper employs two transfer techniques, policy distillation and policy reuse, along with an explicit transfer condition that masks irrelevant knowledge. Experimental results demonstrate that our method achieves a 94.8% success rate in the UAV maneuver scenario, where the direct use of reinforcement learning always fails. The proposed RL framework SubMas-RL is expected to learn an effective policy in complex tasks with sparse rewards.

1. Introduction

The advantages of the Unmanned Aerial Vehicle (UAV) successfully executing 4D (dull, dirty, dangerous, and deep) tasks have attracted wide attention in the defense industry (Choi et al., 2018; Hou et al., 2023). In the denied environments, the increasing potential threats severely threaten the safety of UAVs (Jia et al., 2022; Sun et al., 2015). Despite numerous evasive maneuver strategies for evading an air-to-air missile, finding a maneuver strategy that can evade multiple missiles remains an open problem that limits the survival probability of UAVs (Yang et al., 2020; Yomchinda, 2015). It is urgent to improve maneuver decision-making ability and enhance safety for the UAV.

Reinforcement Learning (RL) has been attached tremendous attention due to its human-level performance in challenging tasks such as Atari games, robotic control, and navigation tasks (Sutton and Barto, 2018; Mnih et al., 2015; Levine et al., 2016). It is a promising way to apply RL algorithms to solve the UAV maneuver decision-making problem (Hu et al., 2022). Regarding long-horizon tasks, the RL agent depends on dense rewards to train and construct long-term decision sequences (Xu et al., 2022; Nagpal et al., 2020). However, reward signals are highly sparse in the UAV maneuver decision-making problem due to the lack of expert experience (Sun et al., 2021). To our best knowledge, dealing with the sparse reward is still an open problem that hinders wider applications of RL algorithms.

Recent developments in RL have improved exploration efficiency in sparse reward tasks (Zhang et al., 2020; Sohn et al., 2018; He et al., 2022). In the setting of hierarchical RL, the task is decomposed into subgoals with finer time resolutions, and the action space is grouped into different scales to form a high-level option that chooses low-level policies (Bacon et al., 2017; Florensa et al., 2017; Eysenbach et al., 2018). These goal-conditioned methods usually assume that each subgoal belongs to different temporal scales, so as to select different policies according to subgoals at each time step (Levy et al., 2017). Unfortunately, in some complex tasks, the agent must handle different subgoals simultaneously, e.g., multiple missiles in the UAV maneuver scenario. Another example is the Super Mario game, where Mario needs to jump across platforms and collect coins while evading Goombas. However, goal-conditioned methods fail to handle these subgoals, which may overlap in temporal scales.

Different from hierarchical RL, curriculum learning in RL aims to construct a set of easier source tasks (a curriculum) inherited from a target task, in order to accelerate learning speed (Narvekar et al., 2020). Narvekar et al. (2016) proposed the concept of task dimension simplification, which simplifies complex tasks with different degrees of freedom and restrictions. For instance, we can remove coins or

* Corresponding author.
E-mail addresses: afeu_hyq@163.com (Y. Hou), afeu_lxl@sina.com (X. Liang), maolonglv@163.com (M. Lv), q.yang@tudelft.nl (Q. Yang), Y.Li-31@tudelft.nl (Y. Li).

https://doi.org/10.1016/j.engappai.2023.106703
Received 8 October 2022; Received in revised form 20 April 2023; Accepted 20 June 2023
Available online xxxx
0952-1976/

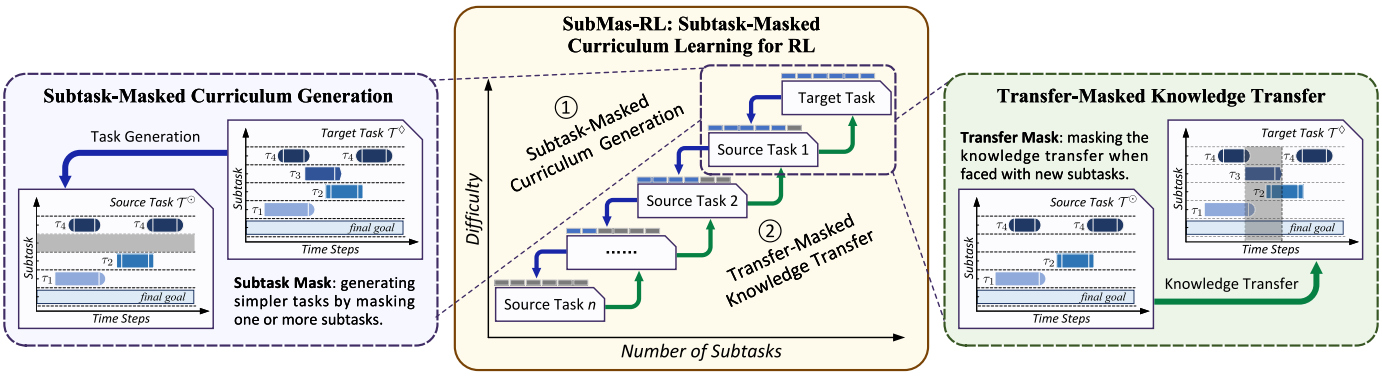


Fig. 1. The overall workflow of training an agent for a target task by SubMas-RL, which is composed of two key components. Subtask-Masked Task Generation (left module) serves to generate a sequenced curriculum by masking one or more subtasks. Transfer-Masked Knowledge Transfer (right module) aims to transfer the knowledge learned from intermediate tasks to accelerate training in the target task.

Goombas in Super Mario games to generate an easier game. In contrast to mutually exclusive subgoals, collecting coins and evading Goombas in Super Mario games can be regarded as subtasks, which are allowed to overlap in temporal scales. This idea provides a new perspective for solving complex tasks which can hardly be decomposed into subgoals. However, the task dimension simplification method proposed in Narvekar et al. (2016) relies on the knowledge of the domain's parameterization and lacks a practically general algorithm.

Drawing inspiration from Narvekar et al. (2016), this paper focuses on a target task composed of a final goal and multiple subtasks. In the UAV maneuver scenario, evading one missile can be considered a subtask, and the final goal is to reach the destination. For curriculum learning, we investigate task generation, task sequencing, and knowledge transfer under the subtask setting. As a first step towards curriculum development, we consider how to automatically generate easier source tasks by removing one or more subtasks from the target task. The removal of subtasks reduces the difficulty while transforming the target task's Markov Decision Process (MDP), including state space and reward functions (Abel, 2022). It is significant to build the connection between tasks and construct inter-task mapping. Then, we consider how to automatically sequence source tasks into a curriculum under the subtask setting. Previous work employs MDP elements as ordering metrics, such as initial state, reward function, and transition distribution (Ivanovic et al., 2019). It is worth noting that the difficulty of source tasks is not explicitly related to the above elements but depends on subtasks. Finally, we consider how to transfer knowledge between tasks with different subtasks. Drawing inspiration from transfer learning, the knowledge encoded into trained source policies can be used to accelerate the learning speed on the target task (Taylor et al., 2007). Since two tasks have different subtask sets and state space, knowledge transfer between tasks is not directly permitted under the subtask setting.

To address these problems, this paper proposes two critical concepts: *subtask mask* and *transfer mask*, which lay a foundation for curriculum generation and knowledge transfer. Specifically, we develop a practical algorithm for the UAV maneuver strategy learning, called the **Sub**task-**Mas**ked curriculum learning for **RL** (SubMas-RL). Fig. 1 depicts the overall workflow of training an agent for a target task by SubMas-RL, including subtask-masked curriculum generation and transfer-masked knowledge transfer. The target task is composed of a final goal g_f and a series of potential subtasks $\{\tau_1, \tau_2, \tau_3, \tau_4\}$. Source task 1 is the result of masking one subtask in the target task, and the task generation is transitive until source task n. After building a sequenced curriculum, the training starts from source task n based on RL algorithms. The trained RL model on the previous task is leveraged to guide the training of the next task. Knowledge transfer should be forbidden when the agent faces a fresh subtask, e.g., τ_3 in Fig. 1. Finally, we obtain a target policy dedicated to the target task. The main contributions of this paper are summarized as follows:

- We proposed a subtask-masked curriculum generation method to generate a simpler task by masking partial subtasks in a complex task while abstracting the subtask-related states and rewards. Task generation and task sequencing are alternatively performed to generate the sequenced curriculum, which allows efficient RL by starting from learning on a simple scenario with one subtask and progressively increasing the number of subtasks.
- We employed policy distillation and policy reuse techniques to enable knowledge transfer from policies trained in source tasks to the target task. In addition, we designed the state-dependent transfer condition to prevent the negative transfer, based on which policy distillation and policy reuse are then combined with adaptive weights to stabilize the training.
- The proposed SubMas-RL algorithm is compared with three typical reinforcement learning algorithms, and results show that our method can enable efficient exploration, achieving a success rate of 94.8% in the UAV maneuver scenario where the direct implementation of reinforcement learning always fails. Moreover, we conducted an ablation study to investigate the effectiveness of each module of the proposed method.

The rest of this paper is organized as follows. Section 2 provides a comprehensive background to the research area. Section 3 presents the SubMas-RL algorithm. Section 4 provides experiment setup and results, and Section 5 provides a critical discussion on results. Section 6 introduces related works, and Section 7 draws the conclusion.

2. Background

In this section, we formalize the standard RL problem and describe how it is typically solved by the deterministic policy gradient algorithm. We then introduce curriculum learning in the context of RL. Finally, we present the definition of the UAV maneuver decision-making problem.

2.1. Reinforcement learning

Consider a standard RL problem in which an agent interacts with a stochastic environment that follows an MDP, in order to maximize cumulative reward. An MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \varrho)$: a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, an initial state distribution $\varrho : \mathcal{S} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. Without prior knowledge about the transition and reward functions, the agent utilizes its policy π_θ to collect a trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, \ldots\}$ for training. Denote the density at state s' after transitioning for t time steps from state s under policy π by $p(s \to s', t, \pi)$. The agent improves its policy π_θ to maximize the expected return for each episode:

$$J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[R(s, a)], \tag{1}$$

where $\rho^{\pi}(s') := \int_S \sum_{t=1}^{\infty} \gamma^{t-1} \rho(s) p\left(s \to s', t, \pi\right) \mathrm{d}s$ is the (improper) discounted state visitation distribution for the policy π (Silver et al., 2014). The state–action value function is defined as the discounted return starting from $(s, a) \in S \times \mathcal{A}$:

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t) \mid s_0 = s, a_0 = a\right]. \tag{2}$$

Given a parameterized deterministic policy $\mu_\theta(s)$ which maps a state to a specific action, the fundamental result underlying this problem is the Deterministic Policy Gradient (DPG) theorem (Silver et al., 2014). The policy gradient of the optimization objective (1) with respect to deterministic policy $\mu_\theta(s)$ is

$$\nabla_\theta J\left(\mu_\theta\right) = \mathbb{E}_{s\sim\rho^{\mu}}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q_\phi(s,a)\big|_{a=\mu_\theta(s)}\right]. \tag{3}$$

where $Q_\phi(s,a)$ is the Q-value function (2) parameterized by ϕ.

To approximate the value function in a stable and robust way, the DDPG algorithm (Lillicrap et al., 2015) integrates the successor of DQN (Mnih et al., 2013) into DPG, including a replay buffer and a target network. In addition, the TD3 algorithm introduces a clipped Double Q-Learning variant to limit overestimation bias by taking the minimum value between a pair of critics (Fujimoto et al., 2018):

$$y = r + \gamma \min_{i=1,2} Q_{\phi_i'}(s', a'). \tag{4}$$

Moreover, the TD3 algorithm deploys a delayed policy update strategy, which effectively eliminates the interplay between the actor and critic updates. In this paper, we utilize the TD3 algorithm in the proposed SubMas-RL algorithm to train deterministic policies.

2.2. Curriculum learning for reinforcement learning

Curriculum learning is a promising way to solve problems that might be too difficult to learn from scratch. In the context of RL, a curriculum can be typically constructed as a sequence of tasks or an ordering structure over experience samples (Narvekar et al., 2020). This paper focuses on task-level curriculum learning that encompasses task generation, task sequencing, and knowledge transfer (Silva and Costa, 2018). Algorithm 1 summarizes the three main steps for building and utilizing a curriculum.

Algorithm 1 Curriculum Generation and Utilization

Input: Target task $\mathcal{T}^{\diamondsuit}$
1 $\boldsymbol{\mathcal{T}}^{\odot} \leftarrow$ createTasks($\mathcal{T}^{\diamondsuit}$) ▷Task generation
2 $\mathbb{C} \leftarrow$ bulidCurriculum($\boldsymbol{\mathcal{T}}^{\odot}, \mathcal{T}^{\diamondsuit}$) ▷Task sequencing
3 $\pi^{\diamondsuit} \leftarrow$ learn($\mathbb{C}, \mathcal{T}^{\diamondsuit}$) ▷Knowledge transfer
Output: Trained policy $\pi^{\diamondsuit}$ for the target task

Given a target (hard) task $\mathcal{T}^{\diamondsuit}$, *task generation* aims to create a set of source (easier) tasks $\boldsymbol{\mathcal{T}}^{\odot}$. Denote the task set by $\boldsymbol{\mathcal{T}} = \{\mathcal{T}^{\diamondsuit}, \boldsymbol{\mathcal{T}}^{\odot}\}$. Each task $\mathcal{T}_i \in \boldsymbol{\mathcal{T}}$ is formalized as an MDP $(S_i, \mathcal{A}_i, \mathcal{P}_i, R_i, \varrho_i)$.

Task sequencing serves to organize the task set $\boldsymbol{\mathcal{T}}$ into a directed graph way (a curriculum $\mathbb{C}$) that specifies the ordering of tasks. A task-level curriculum $\mathbb{C} = (\mathcal{V}, \mathcal{E}, \boldsymbol{\mathcal{T}})$ is a directed acyclic graph, where each vertex $v_i \in \mathcal{V}$ is assigned with a single task $\mathcal{T}_i \in \boldsymbol{\mathcal{T}}$. Each directed edge $\langle v_j, v_k\rangle \in \mathcal{E}$ indicates that task $\mathcal{T}_j$ associated with vertex v_j should be trained on before task $\mathcal{T}_k$ associated with vertex v_k. All directed paths in curriculum $\mathbb{C}$ are absorbed into a single sink node v_t (the target task $\mathcal{T}^{\diamondsuit}$). As the most common structure for task-level curriculum, sequenced curriculum orders the task set as a linear chain. The indegree and outdegree of each vertex in a sequenced curriculum are at most 1.

Knowledge transfer extracts and reuses the knowledge learned from intermediate tasks to another within the curriculum. Transfer learning methods provide an effective way to transfer knowledge across tasks.

The above three steps of curriculum learning can be used to create tasks, order tasks, and solve the target task by progressively increasing the difficulty.

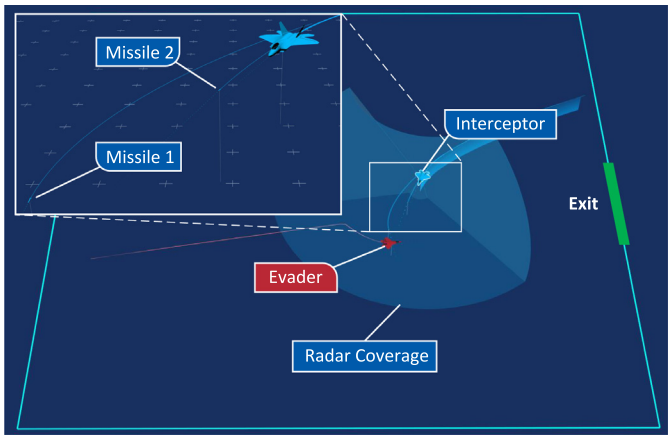


Fig. 2. UAV maneuver scenario where an evader attempts to reach the exit point while evading missiles launched by the interceptor.

2.3. UAV maneuver decision-making problem

The considered scenario is constructed using a top-down design approach, which starts with a high-level objective and decomposes it into smaller sub-objectives until a complete scenario is defined (Crespi et al., 2008). The top-down design approach for the UAV maneuver scenario involves identifying the overall objective of finding a maneuver strategy that allows the evader to reach its destination safely. The sub-objectives include interception of the evader by the interceptor, evasion of missiles by the evader, and reaching the exit point located on the right side of the battlefield. To achieve these sub-objectives, it is necessary to create a comprehensive scenario that includes the evader and interceptor randomly initialized on opposite sides of the battlefield, as shown in Fig. 2. The interceptor is equipped with four mid-range missiles and needs to continuously track down the evader and launch missiles at intervals. The evader attempts to reach the exit point while evading multiple missiles launched by the interceptor. By breaking down the scenario into sub-objectives and defining the necessary components to achieve each sub-objective, a top-down design approach can be used to ensure that the ultimate goal of finding a maneuver strategy for the evader to evade multiple missiles and reach its destination safely is achieved.

The practicality of the considered scenario lies in its potential applications in various real-world scenarios, particularly in military operations such as air combat, wherein the ability of UAV maneuver execution is critical. The proposed framework presents a means to develop effective maneuver strategies, allowing for evading multiple missiles while reaching a destination safely. Furthermore, the trained maneuver strategies can be employed during missions related to surveillance, reconnaissance, or in search and rescue operations, enhancing UAVs' survivability in respective scenarios.

Regular maneuver strategies for evading missiles only consider a single missile and cannot be directly applied to this scenario. A promising approach is to apply RL algorithms in this scenario to train a maneuver strategy. At each time step t, the evader obtains the observation s_t, including the state of the evader, interceptor, and missiles. Then, the evader makes the decision a_t to control the flight, leading to the transition from the current state to the next. The evader continually improves its policy during training by maximizing cumulative rewards. Next, we formalize the UAV maneuver decision-making problem as an MDP.

State Space. Following the definition in Hu et al. (2022), we integrate all the system states into a high-level representation, including the dual UAVs' states and missiles' states: $s = [x, y, \psi, l, \theta_{\mathrm{AA}}, \theta_{\mathrm{ATA}}, \phi_{\mathrm{Exit}}, r_{\mathrm{m}}^1, \phi_{\mathrm{m}}^1, \ldots, r_{\mathrm{m}}^4, \phi_{\mathrm{m}}^4]$. (x, y) is the position of the evader, ψ is the yaw angle of the evader, l is the distance between dual UAVs, θ_{AA} is the aspect

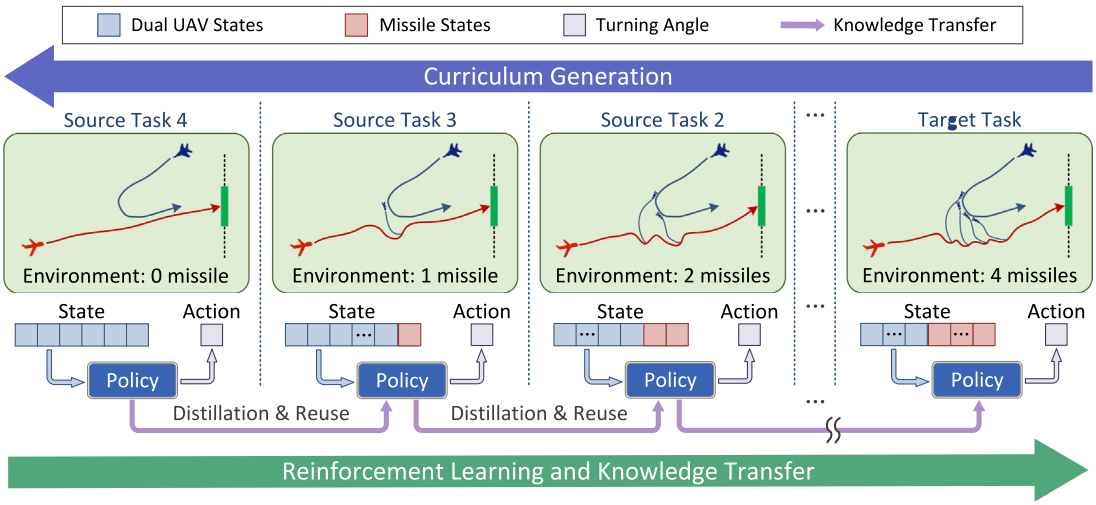


Fig. 3. An overview of the SubMas-RL applied to a UAV maneuver task. The sequenced curriculum is generated by masking one or more missiles in the target task. Policy distillation and policy reuse are employed to transfer knowledge between tasks.

angle between the line of sight (LOS) and the tail of the interceptor, θ_{ATA} is the antenna train angle between the nose of the evader and the LOS, and ϕ_{Exit} is the azimuth angle of the exit's central point. r_{m}^i is the distance between evader and ith missile, and ϕ_{m}^i is the azimuth angle of ith missile. Note that the missile states are unavailable until the interceptor launches the missile. Since the missile states are defined with the constraints $r_{\text{m}}^i \geq 0$ and $\phi_{\text{m}}^i \in (-\pi, \pi]$, choosing the default state vector as $[-1, -\pi]$ guarantees that it does not fall within the domain of valid state values. Thus, we assign default state vector $[r_{\text{m}}^i, \phi_{\text{m}}^i] = [-1, -\pi]$ and fill the ground truth once the missile is launched. This can help to avoid potential issues related to numerical instabilities that may arise if the default state vector is too close to the valid state values.

Action Space. The evader's action is the turning angle between the expected heading and yaw angle, chosen in a continuous space: $a = \Delta\psi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. We employ three-degree-of-freedom UAV dynamics in the simulation platform, and the built-in controller converts the expected turning angle into thrust, elevator, and baking angle.

Reward Function. Previous works regarding similar scenarios rely on dense rewards tuned by human experts, where the missile is simplified as an attack zone (McGrew et al., 2010; Hu et al., 2022). Unlike previous works, we do not simplify the missile and consider the proportional-navigation guidance model of missiles (Yang and Yang, 1996). Due to the lack of domain experience, it is impractical to design a dense reward function that considers multiple missiles. Thus, we design a sparse reward function based on the key events. The evader is rewarded a 400 bonus for reaching the exit, a 50 bonus for evading one missile's attack, a −400 bonus for being shot down, and a −0.1 bonus per time step.

Termination condition. The environment is terminated when the evader reaches the exit point or is shot down. The maximum episode length is 1200 steps, and one step is equivalent to one second in reality. To avoid negative behavior, the evader is considered shot down if it remains more than 10 steps outside the battlefield.

Due to sparse rewards and long-horizon decision sequences for evading missiles, it is difficult to train this task by directly implementing standard RL algorithms, especially when multiple missiles exist. In this paper, we propose the SubMas-RL algorithm to facilitate efficient RL in the UAV maneuver scenario by starting from learning on a simple scenario with one missile and progressively increasing the number of missiles.

Fig. 3 depicts the overview of the SubMas-RL applied to the UAV maneuver task. The training aims to obtain a policy dedicated to the target task, which involves evading 4 missiles and reaching the exit (green square on the right). Each missile is considered a subtask, and the source tasks are generated by masking one or more missiles. Source task 4 is a naive navigation task without missiles, which can be solved by an expert policy. The training starts from source task 3 by leveraging the expert policy. The policy is trained by RL algorithms with the techniques of policy distillation and policy reuse. Finally, we train the evader on the target task and obtain the target policy. More details about curriculum generation and knowledge transfer will be introduced in Sections 3.2 and 3.3.

3. Methodology

The UAV maneuver decision-making problem is a long-horizon task with sparse rewards. The agent (evader) aims to reach a final goal (exit) while dealing with a series of subtasks (missiles). Due to parallel subtasks and sparse rewards, the target task is too difficult to be directly trained by standard RL algorithms. Drawing from curriculum learning, gradually increasing the task difficulty can facilitate progressive learning and improve performance in the target task. Thus, we employ the curriculum learning framework to train the RL model more efficiently. In this section, we first define the concept of subtask mask and transfer task. Then, we propose two critical modules: subtask-masked curriculum generation and transfer-masked knowledge transfer, based on which we propose a practical training algorithm, called **Sub**task-**Mas**ked curriculum learning for **RL** (SubMas-RL) algorithm.

3.1. Definitions

According to Silva and Costa (2018), curriculum learning involves task generation, task sequencing, and knowledge transfer. Under the subtask setting, two questions naturally arise:

(i) How to automatically generate source tasks and build a sequenced curriculum using subtasks?

(ii) How to transfer knowledge among the sequenced curriculum where each task has different subtask sets and state spaces?

We first take Super Mario games as an example to discuss the above questions. The final goal of Super Mario is to reach the destination; the subtasks are collecting coins and evading Goombas. Removing Goombas in this game can reduce the difficulty and generate easier games. Thus, a feasible solution for question (i) is to remove partial subtasks. The agent can be trained in a Goombas-free game, and the knowledge gained can be transferred to guide the training on the full game. Note however, that the knowledge gained may be imperfect to deal with the full game, as it does not include the new elements

introduced by Goombas. For question (ii), it is significant to determine the condition under which knowledge transfer can provide the correct guidance. To provide a formal answer to the above questions, we introduce two concepts, subtask mask and transfer mask.

Definition 1 (*Subtask Mask*)**.** Given a target task $\mathcal{T}^{\Diamond}$ with a final goal g_f and multiple subtasks $\boldsymbol{\tau}^{\Diamond} = \{\tau_1, \tau_2, \ldots, \tau_n\}$, subtask mask serves to mask one or more subtasks $\boldsymbol{\tau} \subseteq \boldsymbol{\tau}^{\Diamond}$ while removing subtask-related states and rewards, in order to generate a smaller and easier (source) task $\mathcal{T}^{\odot}$.

The *subtask mask* is used to generate a source task with fewer subtasks automatically. As shown in Fig. 1 (the left module), source task 1 is generated from the target task by masking one subtask, e.g., τ_3 in this example. Meanwhile, state abstraction reduces the task complexity by eliminating states and rewards related to the masked subtasks (Abel, 2022). Section 3.2.1 formulates the subtask-masked *task generation* method. Furthermore, these source tasks can be sequenced into a task-level curriculum $\mathbb{C}$. We introduce the implementation of subtask-masked *task sequencing* in Section 3.2.2.

Knowledge transfer among the curriculum leverages prior knowledge to guide exploration and facilitate efficient RL training. However, knowledge is not always allowed to be transferred across tasks due to different subtask sets. As shown in Fig. 1 (the right module), knowledge transfer from source task 1 to the target task is prevented when the agent faces a fresh subtask τ_3. For instance, in Super Mario games, the agent trained in a Goombas-free task dedicates to jumping across platforms and collecting coins. In a new task involving Goombas, the knowledge about platforms and coins still works, but knowledge transfer from a Goombas-free task to the full game should be forbidden when Mario faces Goombas.

Definition 2 (*Transfer Mask*)**.** Given a source task, which is generated by masking subtasks $\boldsymbol{\tau}$ in the target task, knowledge transfer from source task to target task should be prevented (masked) when the agent faces any fresh subtask $\tau \in \boldsymbol{\tau}$ which is not experienced in the source task.

The *transfer mask* filters useless or even negative knowledge transfer, thus enhancing transfer efficiency. The agent can adapt to a new environment quickly by leveraging useful knowledge obtained from the imperfect policy. In terms of exploration, when the policy fails to provide correct guidance, the agent will then independently explore the environment. The transfer-masked *knowledge transfer* method is presented in Section 3.3.

3.2. Subtask-masked curriculum generation

We discuss how to construct a task-level sequenced curriculum under the subtask setting. Specifically, we formalize the task generation and task sequencing methods.

3.2.1. Task generation

In this subsection, we contribute a task generation method to generate source tasks from a target task (as a createTasks function in Algorithm 1, line 1). Each source task has different subtask sets as well as different difficulties.

Fig. 4 provides an overview of the subtask-masked task generation process, which involves subtask mask and state abstraction. To generate the source task $\mathcal{T}^{\odot}$ from the target task $\mathcal{T}^{\Diamond}$, the subtask mask serves to mask partial subtasks within the target task. Additionally, state abstraction is used to eliminate irrelevant states from the target task.

Given a target task $\mathcal{T}^{\Diamond} = \left(\mathcal{S}^{\Diamond}, \mathcal{A}^{\Diamond}, \mathcal{P}^{\Diamond}, R^{\Diamond}, \varrho^{\Diamond}, \boldsymbol{\tau}^{\Diamond}\right)$ where $\boldsymbol{\tau}^{\Diamond} = \{\tau_1, \tau_2, \cdots, \tau_n\}$ is the subtask set. We apply the subtask mask in the target task to generate a source task $\mathcal{T}^{\odot} = \left(\mathcal{S}^{\odot}, \mathcal{A}^{\odot}, \mathcal{P}^{\odot}, R^{\odot}, \varrho^{\odot}, \boldsymbol{\tau}^{\odot}\right)$, where $\boldsymbol{\tau}^{\odot}$ is the subtask set in the source task, and $\boldsymbol{\tau}^{\odot} \subset \boldsymbol{\tau}^{\Diamond}$. The masked subtasks are defined as $\boldsymbol{\tau}_{\odot}^{\Diamond} = \boldsymbol{\tau}^{\Diamond} - \boldsymbol{\tau}^{\odot}$.

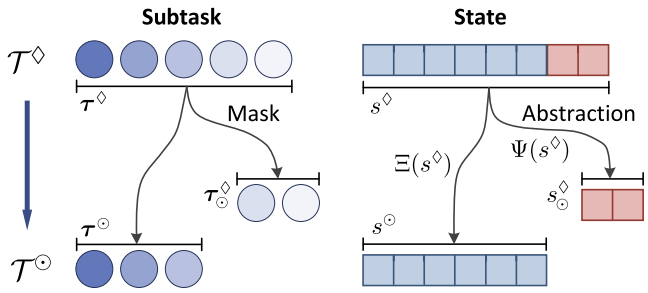


Fig. 4. The overview of subtask-masked task generation. The source task $\mathcal{T}^{\odot}$ is the result of applying subtask mask and state abstraction in the target task $\mathcal{T}^{\Diamond}$.

We then construct an inter-task mapping between the target and source tasks (Abel, 2022). We assume the subtask mask does not alter the action space, i.e., $\mathcal{A}^{\Diamond} = \mathcal{A}^{\odot}$. The state vector of the target task is denoted by $s^{\Diamond} = \left[s_1, s_2, \ldots, s_{d^{\Diamond}}\right]$, where $d^{\Diamond} = \dim(\mathcal{S}^{\Diamond})$. In the absence of the masked subtasks $\boldsymbol{\tau}_{\odot}^{\Diamond}$, we remove irrelevant states from the target task (Abel, 2022). Thus, the state vector $s^{\Diamond}$ is decomposed into two parts (see Fig. 4):

$$\begin{aligned} s^{\Diamond} &= \left[s^{\odot}, s_{\odot}^{\Diamond}\right], \\ s^{\odot} &= \left[s_1, s_2, \ldots, s_{d^{\odot}}\right], \\ s_{\odot}^{\Diamond} &= \left[s_{d^{\odot}+1}, \ldots, s_{d^{\Diamond}}\right], \end{aligned} \tag{5}$$

where $s_{\odot}^{\Diamond} \in \mathcal{S}_{\odot}^{\Diamond}$ which is the masked state space only related with the masked subtasks $\boldsymbol{\tau}_{\odot}^{\Diamond}$. Such inter-task mappings $\Xi : \mathcal{S}^{\Diamond} \to \mathcal{S}^{\odot}$ and $\Psi : \mathcal{S}^{\Diamond} \to \mathcal{S}_{\odot}^{\Diamond}$ for state space, take a state from $\mathcal{S}^{\Diamond}$ and map it into a state in $\mathcal{S}^{\odot}$ and $\mathcal{S}_{\odot}^{\Diamond}$:

$$s^{\odot} = \Xi(s^{\Diamond}), s_{\odot}^{\Diamond} = \Psi(s^{\Diamond}). \tag{6}$$

Following the definition of state abstraction in Abel (2022), the subtask mask does not alter the initial state distribution and transition probability distribution in the shared state space of $\mathcal{T}^{\odot}$ and $\mathcal{T}^{\Diamond}$, i.e., $\forall s^{\Diamond}, s'^{\Diamond} \in \mathcal{S}^{\Diamond}, a \in \mathcal{A}$,

$$\varrho^{\odot}\left(\Xi\left(s^{\Diamond}\right)\right) = \int_{\mathcal{S}_{\odot}^{\Diamond}} \varrho^{\Diamond}(s^{\Diamond}) \mathrm{d}\Psi(s^{\Diamond}), \tag{7}$$

$$\mathcal{P}^{\odot}(\Xi(s'^{\Diamond}) | \Xi(s^{\Diamond}), a) = \int_{\mathcal{S}_{\odot}^{\Diamond}} \int_{\mathcal{S}_{\odot}^{\Diamond}} \mathcal{P}^{\Diamond}\left(s'^{\Diamond} | s^{\Diamond}, a\right) \mathrm{d}\Psi(s^{\Diamond}) \mathrm{d}\Psi(s'^{\Diamond}). \tag{8}$$

We assume the reward function in the target task can be linearly decomposed according to subtasks:

$$R^{\Diamond}\left(s^{\Diamond}, a\right) = R^{g_f}\left(s^{\Diamond}, a\right) + \sum_{\tau_i \in \boldsymbol{\tau}^{\Diamond}} R^{\tau_i}\left(s^{\Diamond}, a\right), \tag{9}$$

where $R^{g_f}(\cdot,\cdot)$ and $R^{\tau_i}(\cdot,\cdot)$ are the reward function of the final goal g_f and subtask τ_i, respectively. Since the subtasks $\boldsymbol{\tau}_{\odot}^{\Diamond}$ are masked in the source task, the corresponding reward functions should also be removed from Eq. (9):

$$R^{\odot}\left(\Xi(s^{\Diamond}), a\right) = R^{\Diamond}\left(s^{\Diamond}, a\right) - R_{\odot}^{\Diamond}\left(s^{\Diamond}, a\right), \tag{10}$$

where

$$R_{\odot}^{\Diamond}\left(s^{\Diamond}, a\right) = \sum_{\tau_i \in \boldsymbol{\tau}_{\odot}^{\Diamond}} R^{\tau_i}\left(s^{\Diamond}, a\right). \tag{11}$$

Algorithm 2 summarizes the subtask-masked task generation method. Taking different masked subtasks as input, Algorithm 2 is used to generate various source tasks which compose the source task set $\mathcal{T}^{\odot}$.

Next, we take a robot navigation task as an example and discuss the shared knowledge between the target task and the source task. In a full navigation task $\mathcal{T}^{\Diamond}$, an agent has access to the position of the final goal and sensor observations of random obstacles. The

Algorithm 2 Subtask-Masked Task Generation

Input: Target task $\mathcal{T}^{\diamondsuit}$, masked subtasks $\boldsymbol{\tau}_{\odot}^{\diamondsuit}$

$\boldsymbol{\tau}^{\odot} \leftarrow \boldsymbol{\tau}^{\diamondsuit} - \boldsymbol{\tau}_{\odot}^{\diamondsuit}$

$\mathcal{A}^{\odot} \leftarrow \mathcal{A}^{\diamondsuit}$

$S^{\odot}, S_{\odot}^{\diamondsuit} \leftarrow \text{stateSpace}(S^{\diamondsuit}, \boldsymbol{\tau}^{\odot}, \boldsymbol{\tau}_{\odot}^{\diamondsuit})$ ▷Eq. (5)

$\Xi, \Psi \leftarrow \text{interTaskMapping}(S^{\diamondsuit}, S^{\odot}, S_{\odot}^{\diamondsuit})$ ▷Eq. (6)

$\rho^{\odot} \leftarrow \text{initState}(\rho^{\diamondsuit}, \Xi, \Psi)$ ▷Eq. (7)

$\mathcal{P}^{\odot} \leftarrow \text{transitionFunction}(\mathcal{P}^{\diamondsuit}, \Xi, \Psi)$ ▷Eq. (8)

$R^{\odot} \leftarrow \text{rewardFunction}(R^{\diamondsuit}, \Xi, \Psi)$ ▷Eq. (10)

Output: Source task $\mathcal{T}^{\odot}$, inter-task mapping Ξ, Ψ

agent attempts to reach the final goal without collision. By removing obstacles and masking sensor observations, this task can be reduced to a more straightforward task $\mathcal{T}^{\odot}$. Obviously, the shared knowledge between two tasks is how to navigate to the final goal. In terms of knowledge transfer, when the agent in $\mathcal{T}^{\diamondsuit}$ does not detect any obstacle, it is allowed to utilize the shared knowledge regarding navigating to the final goal. Therefore, the critical criterion for sharing knowledge is whether the agent detects obstacles or, generally speaking, whether new subtasks appear. The observation of obstacles stored in $s_{\odot}^{\diamondsuit}$ can serve as a criterion. When the agent detects obstacles, $s_{\odot}^{\diamondsuit}$ returns the ground truth of obstacles; otherwise, it returns a default vector.

Drawing from the above example, we hereby present a criterion for sharing knowledge. Define $\widetilde{s}_{\odot}^{\diamondsuit}$ as a default state vector of $s_{\odot}^{\diamondsuit}$. The condition $s_{\odot}^{\diamondsuit} = \widetilde{s}_{\odot}^{\diamondsuit}$ means that the masked subtasks $\boldsymbol{\tau}_{\odot}^{\diamondsuit}$ do not appear in the target task. Correspondingly, the reward that relates to these subtasks is equal to zero:

$$R_{\odot}^{\diamondsuit}\left(s^{\diamondsuit}, a\right) \equiv 0, \text{ s.t. } \Psi\left(s^{\diamondsuit}\right) = \widetilde{s}_{\odot}^{\diamondsuit}, \tag{12}$$

According to Eqs. (10), (11) and (12), the rewards of target and source tasks are identical when $\Psi\left(s^{\diamondsuit}\right) = \widetilde{s}_{\odot}^{\diamondsuit}$. The shared rewards ensure that the knowledge learned in the source task is reusable in the target task. On the contrary, when $\Psi\left(s^{\diamondsuit}\right) \neq \widetilde{s}_{\odot}^{\diamondsuit}$, one or more subtasks in $\boldsymbol{\tau}_{\odot}^{\diamondsuit}$ appear, and $R_{\odot}^{\diamondsuit}\left(s^{\diamondsuit}, a\right) \neq 0$. This indicates that the knowledge fails to share under this condition, and knowledge transfer should be prevented. More details about the transfer condition are introduced in Section 3.3.1.

3.2.2. Task sequencing

A subtask set $\boldsymbol{\tau}^{\diamondsuit} = \{\tau_1, \tau_2, \dots \tau_n\}$ has $2^n - 1$ non-void proper subsets. That is, we can generate $2^n - 1$ source tasks by masking different subtasks. However, not all of these source tasks are necessary for building a curriculum. We hereby propose a method to build a task-level sequenced curriculum that is composed of n source tasks and the target task (as a bulidCurriculum function in Algorithm 1, line 2). Our sequencing method orders the source tasks according to the difficulty, which depends on the number of involved subtasks. Specifically, the source tasks with fewer subtasks are prioritized for training before those with more subtasks.

Fig. 5 presents a schematic of building a sequenced curriculum from a target task with 4 subtasks. Each subtask $\tau_i \in \boldsymbol{\tau}^{\diamondsuit}$ is manually ordered by humans according to its complexity. For instance, in Super Mario games, the player usually finds that evading a Goomba is more complex than collecting coins. Taking the most complex subtask τ_1 as input, a source task $\mathcal{T}_1^{\odot}$ is generated from $\mathcal{T}^{\diamondsuit}$ (by Algorithm 2). These two tasks are naturally ordered because of the fewer subtasks in $\mathcal{T}_1^{\odot}$. As such, the next step starts from $\mathcal{T}_1^{\odot}$ and generates the source task $\mathcal{T}_2^{\odot}$, and so on. All tasks are integrated into a sequenced curriculum $\mathbb{C}$.

Algorithm 3 summarizes the subtask-masked task sequencing method. Each source task is generated from the last source task by masking one subtask. Each directed edge $\langle v_i, v_{i-1}\rangle \in \mathcal{E}$ indicates

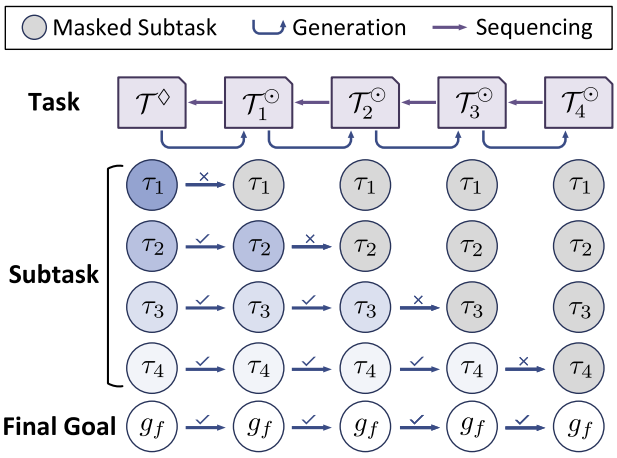


Fig. 5. Building a task-level sequenced curriculum from a target task $\mathcal{T}^{\diamondsuit}$ with 4 subtasks. Task generation serves to create a new source task by masking one subtask. Task sequencing specifies the ordering of tasks and finally builds a curriculum as a linear chain.

Algorithm 3 Subtask-Masked Task Sequencing

Input: Target task $\mathcal{T}^{\diamondsuit}$

Input: Ordered subtask set $\boldsymbol{\tau}^{\diamondsuit} = \{\tau_1, \tau_2, \cdots \tau_n\}$

Initialize: $v_0 \leftarrow \mathcal{T}^{\diamondsuit}, \mathcal{V} \leftarrow \{v_0\}, \mathcal{E} \leftarrow \emptyset, \mathcal{T} \leftarrow \{\mathcal{T}^{\diamondsuit}\}$

$\mathcal{T}_0^{\odot} \leftarrow \mathcal{T}^{\diamondsuit}$

for $i = 1, 2, \cdots, n$ **do**

  $\mathcal{T}_i^{\odot} \leftarrow \text{taskGenaration}(\mathcal{T}_{i-1}^{\odot}, \tau_i)$ ▷Algorithm 2

  $v_i \leftarrow \mathcal{T}_i^{\odot}$ ▷Create a vertex

  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{T}_i^{\odot}\}$ ▷Add a source task

  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_i\}$ ▷Add a vertex

  $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle v_i, v_{i-1}\rangle\}$ ▷Add a directed edge

end

Output: A sequenced curriculum $\mathbb{C} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$

that task $\mathcal{T}_i^{\odot}$ should be trained on before task $\mathcal{T}_{i-1}^{\odot}$. The output is a sequenced curriculum that orders the task set as a linear chain. This method ensures each source task has fewer subtasks than the last one. Furthermore, it should be noted that adjacent tasks demonstrate variation in only one subtask. This suggests that sufficient subtask knowledge can be effectively transferred between such adjacent tasks.

We may also consider more automatic sequencing strategies, such as using a subtask set to calculate transfer potential (Svetlik et al., 2017; Silva and Costa, 2018). However, we leave it as future work and focus on how to transfer the knowledge learned from intermediate tasks to improve performance in the target task.

3.3. Transfer-masked knowledge transfer

In this subsection, we propose the transfer-masked knowledge transfer method (as a learn function in Algorithm 1, line 3). We present the transfer condition under the subtask setting and employ policy distillation and policy reuse techniques to enable knowledge transfer.

3.3.1. Transfer condition

A task-level sequenced curriculum is generated by Algorithm 3 to solve a target task that is too difficult to learn from scratch. We expect that the knowledge learned from the source tasks will help the agent explore the target task. Note however, that the policy trained on one task cannot be directly transferred to another since any two tasks have different subtask sets, state spaces, and reward functions. When transferring knowledge from a less related scenario, it may inversely

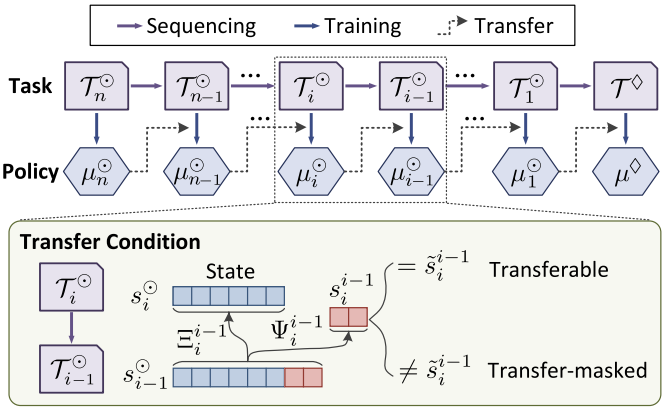


Fig. 6. The training process and transfer condition for SubMas-RL.

hurt the target performance, a phenomenon known as negative transfer (Wang et al., 2019, 2022). A feasible solution is to introduce a transfer condition to filter out irrelevant knowledge transfer (Hu et al., 2015; Liang et al., 2022). Next, we describe the training process and present the condition for knowledge transfer.

Fig. 6 depicts the training process and transfer condition for SubMas-RL. We can train a policy or design an expert policy for the easiest task $\mathcal{T}_n^{\odot}$. Then, we leverage the policy $\mu_n^{\odot}$ to guide the training in next task $\mathcal{T}_{n-1}^{\odot}$ and produce a new policy $\mu_{n-1}^{\odot}$. As such, the policy $\mu_i^{\odot}$ trained in $\mathcal{T}_i^{\odot}$ is utilized to guide the training in $\mathcal{T}_{i-1}^{\odot}$, and so on. The transfer condition in Fig. 6 means that the knowledge transfer should be masked when the agent faces a new subtask. Finally, we solve the target task $\mathcal{T}^{\Diamond}$ by leveraging the policy $\mu_1^{\odot}$ trained in $\mathcal{T}_1^{\odot}$, and obtain the target policy $\mu^{\Diamond}$.

The transfer condition is whether the agent faces a new subtask. Fig. 6 shows the transfer condition from the state perspective. Denote $\Xi_i^{i-1}(\cdot)$ and $\Psi_i^{i-1}(\cdot)$ as inter-task mappings from $\mathcal{T}_{i-1}^{\odot}$ to $\mathcal{T}_i^{\odot}$. Following the definition in (12), we define the transfer condition as

$$\Psi_i^{i-1}\left(s_{i-1}^{\odot}\right) = \tilde{s}_i^{i-1}, \tag{13}$$

where $\tilde{s}_i^{i-1}$ is a default state vector which indicates that the new subtask is not detected in $\mathcal{T}_{i-1}^{\odot}$. When the transfer condition (13) holds true, the knowledge transfer is permitted. On the other hand, if $\Psi_i^{i-1}\left(s_{i-1}^{\odot}\right) \neq \tilde{s}_i^{i-1}$, the agent faces the new subtask, and the knowledge transfer is prevented (masked). The transfer condition serves to filter negative transfer and improve sample efficiency.

3.3.2. Policy distillation

Consider any two adjacent tasks $\mathcal{T}_i^{\odot}$ and $\mathcal{T}_k^{\odot}(k = i-1)$ within the curriculum. Due to different state spaces, the policy cannot be directly transferred from $\mathcal{T}_i^{\odot}$ to $\mathcal{T}_k^{\odot}$. Policy distillation is an effective method for transferring knowledge from one or more policies to an untrained network (Rusu et al., 2015; Li et al., 2023; Zhang et al., 2022). With the inter-task mapping functions Ξ_i^k and Ψ_i^k, policy distillation can work for most standard RL methods by regularizing the policy $\mu_k^{\odot}$ towards the trained policy $\mu_i^{\odot}$ (which we also call the guide policy). With the guide policy $\mu_i^{\odot}$ fixed, we provide an auxiliary reward $D(s,a)$ in $\mathcal{T}_k^{\odot}$ and define a new objective to be optimized, i.e.,

$$\max_{\mu_k^{\odot}} \mathbb{E}_{s\sim\rho^{\mu}}\left[R_k^{\odot}(s,a) + \alpha D(s,a)|_{a=\mu_k^{\odot}(s)}\right], \tag{14}$$

where

$$D(s,a) = -\left\|a - \mu_i^{\odot}\left(\Xi_i^k(s)\right)\right\|_2 \mathbb{1}\left(\Psi_i^k(s) = \tilde{s}_i^k\right). \tag{15}$$

To encourage imitation, the auxiliary reward D measures the negative Euclidean distance between the actions calculated by the policy $\mu_i^{\odot}$ and $\mu_k^{\odot}$. The mapping function Ξ_i^k ensures the state s is mapped into the state space in $\mathcal{T}_i^{\odot}$. The indicator function $\mathbb{1}(\cdot)$ indicates whether the transfer condition holds true. When the transfer condition holds true, the auxiliary reward works, such that the agent (the policy $\mu_k^{\odot}$) learns to imitate the guide policy $\mu_i^{\odot}$.

The weight α in (14) determines the strength of policy distillation. We employ an adaptive weight to find a trade-off between imitation and exploration (Hu et al., 2023; Yang and Spaan, 2023). The goal is to imitate the guide policy more with the larger α in the beginning. Meanwhile, if the agent can gradually find a trajectory to the final goal, the smaller α eliminates the influence of the guide policy. We define the adaptive weight α as

$$\alpha = 1 - \beta, \tag{16}$$

where β reflects the success rate of reaching the final goal g_f. To provide a stable objective during training, β is fixed within one epoch and updated at the end of the epoch. When the agent can always reach the final goal, β is close to 1, such that the influence of α fades away. As a consequence, our method still solves the original problem.

3.3.3. Policy reuse

Our approach adopts the policy reuse strategy that directly leverages a guide policy to sample trajectories, which facilitates rapid adaptation to a new task (Rosman et al., 2016; Fernández and Veloso, 2006). This strategy leads to better initial trajectories and improves the *jump-start* by providing a strong initial point for the learning algorithm (Taylor and Stone, 2007; Yang et al., 2022). The guide policy can take the form of a rule-based policy, expert policy, or well-trained policy (Ayeelyan et al., 2022).

We first present a composite policy μ_c, which is a mixture of the guide policy $\mu_i^{\odot}$ and the training policy $\mu_k^{\odot}$:

$$\mu_c(s_t) = \begin{cases} \mu_i^{\odot}\left(\Xi_i^k(s_t)\right), & \text{if } \Psi_i^k(s_t) = \tilde{s}_i^k, \\ \mu_k^{\odot}(s_t), & \text{otherwise.} \end{cases} \tag{17}$$

When the transfer condition (13) holds, the composite policy (17) directly reuses the guide policy $\mu_i^{\odot}$ to produce an action. However, the guide policy might dominate during training. To find a trade-off between exploration and reuse, we propose a behavior policy μ_b which follows an adaptive ϵ-greedy strategy:

$$\mu_b(s_t) = \begin{cases} \mu_c(s_t), & \text{with probability } \epsilon, \\ \mu_k^{\odot}(s_t), & \text{with probability } 1-\epsilon. \end{cases} \tag{18}$$

Similar to (16), the adaptive probability ϵ varies with the success rate β:

$$\epsilon = \epsilon_0(1-\beta), \tag{19}$$

where $\epsilon_0 < 1$ is the initial probability. With a low success rate β, the behavior policy tends to choose the composite policy μ_c at the beginning. As the success rate β grows, the probability ϵ descends, such that the agent reduces the dependence on the guide policy and collects samples independently.

To stabilize the training, we clip the update of adaptive weights (16) and (19) into a small range:

$$|\alpha_{epoch} - \alpha_{epoch-1}| \le \delta_{\alpha}, \tag{20}$$

$$|\epsilon_{epoch} - \epsilon_{epoch-1}| \le \delta_{\epsilon}, \tag{21}$$

3.4. Training algorithm

Next, we integrate knowledge transfer strategies into an RL algorithm. Since we leverage a behavior policy to sample trajectories, our method is only compatible with off-policy RL algorithms. The auxiliary reward (15) is designed for the deterministic policy and continuous action space. Thus, we employ the TD3 algorithm (Fujimoto et al., 2018).

To transfer the knowledge from $\mathcal{T}_i^{\odot}$ to $\mathcal{T}_k^{\odot}(k = i-1)$, we integrate knowledge transfer strategies into TD3 algorithm (see Algorithm 4).

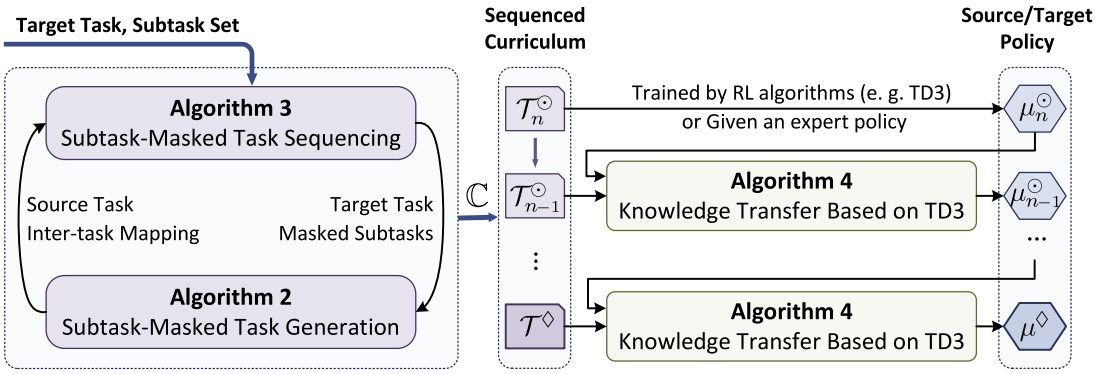


Fig. 7. The flow chart of SubMas-RL algorithm.

Algorithm 4 Knowledge Transfer Based on TD3

Input: Guide policy $\mu_i^{\odot}$, task $\mathcal{T}_k^{\odot}(k = i-1)$
1 **Initialize:** Q_{ϕ_1}, Q_{ϕ_2}, μ_θ, $Q_{\phi_1'}$, $Q_{\phi_2'}$, $\mu_{\theta'}$
2 **Initialize:** $\beta \leftarrow 0$, $\alpha \leftarrow 1$, $\epsilon \leftarrow \epsilon_0$, and $\mathcal{D} \leftarrow \emptyset$
3 **for** *each iteration* **do**
4 **for** *each time step* **do**
5 $a_t \leftarrow \mu_b(s_t) + \delta$, $\delta \sim \mathcal{N}(0, \sigma)$ ▷Policy reuse (18)
6 $r_t \leftarrow R_k^{\odot}(s_t, a_t)$
7 $r_t^d \leftarrow D(s_t, a_t)$ ▷Policy distillation (15)
8 $s_{t+1} \sim \mathcal{P}_k^{\odot}(\cdot|s_t, a_t)$
9 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, r_t^d, s_{t+1})\}$
10 **end**
11 **for** *each gradient step* **do**
12 Sample K transitions (s, a, r, r^d, s') from $\mathcal{D}$
13 $\widetilde{a} \leftarrow \mu_{\theta'}(s') + \delta$, $\delta \sim \text{clip}\left(\mathcal{N}(0, \widetilde{\sigma}), -c, c\right)$
14 $y \leftarrow (r + \alpha r^d) + \gamma \min_{i=1,2} Q_{\phi i'}(s', \widetilde{a})$ ▷Eq. (4)
15 $\phi_i \leftarrow \text{argmin}_{\phi_i} \frac{1}{K} \sum \left(y - Q_{\phi_i}(s, a)\right)^2$
16 **if** *update policy* **then**
17 Update θ by the deterministic policy gradient:
18 $\nabla_\theta J(\theta) = \frac{1}{K} \sum \nabla_a Q_{\phi_1}(s, a)\big|_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s)$ ▷Eq. (3)
19 $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
20 $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
21 **end**
22 **end**
23 $\beta \leftarrow$ success rate of reaching the final goal
24 $\alpha \leftarrow \text{clip}\left(1-\beta, \alpha - \delta_\alpha, \alpha + \delta_\alpha\right)$ ▷Eq. (16), (20)
25 $\epsilon \leftarrow \text{clip}\left(\epsilon_0(1-\beta), \epsilon - \delta_\epsilon, \epsilon + \delta_\epsilon\right)$ ▷Eq. (19), (21)
26 **end**
Output: Optimized parameters θ^* for μ_θ

Algorithm 5 SubMAS-RL

Input: Target task $\mathcal{T}^{\Diamond}$
Input: Ordered subtask set $\boldsymbol{\tau}^{\Diamond} = \{\tau_1, \tau_2, \cdots \tau_n\}$
1 $\mathbb{C} \leftarrow$ buildCurriculum$(\mathcal{T}^{\Diamond}, \boldsymbol{\tau}^{\Diamond})$ ▷Algorithm 3
2 Select the task $\mathcal{T}_n^{\odot}$ from $\mathbb{C}$
3 $\mu_n^{\odot} \leftarrow$ training on $\mathcal{T}_n^{\odot}$ or given an expert policy
4 **for** $i = n, \cdots, 1$ **do**
5 $k \leftarrow i - 1$
6 Choose the task $\mathcal{T}_k^{\odot}$ from $\mathbb{C}$
7 $\mu_k^{\odot} \leftarrow$ knowledgeTransfer$(\mathcal{T}_k^{\odot}, \mu_i^{\odot})$ ▷Algorithm 4
8 **end**
9 $\mu^{\Diamond} \leftarrow \mu_0^{\odot}$
Output: The target policy $\mu^{\Diamond}$ for $\mathcal{T}^{\Diamond}$

The policy $\mu_i^{\odot}$ trained in $\mathcal{T}_i^{\odot}$ serves as a guide policy. For each step, the agent interacts with the environment to collect trajectories using the behavior policy μ_b (Line 5). The reward is a mixture of the original reward (Line 6) and auxiliary reward (Line 7). The actor and critics are updated in terms of the TD3 algorithm (Lines 12–20). We count the average success rate β (Line 23) for each epoch and update the adaptive weights α and ϵ (Lines 24–25). In summary, Algorithm 4 serves to train the agent in $\mathcal{T}_k^{\odot}$ by leveraging the knowledge embedded in $\mu_i^{\odot}$.

Fig. 7 depicts the flow chart of SubMas-RL algorithm. Given a target task and subtask set, the first module generates a sequenced curriculum by alternately conducting task generation and task sequencing. For the easiest task $\mathcal{T}_n^{\odot}$, we can train a policy by RL algorithms (e.g., TD3) or design a rule-based (expert) policy. Then, source tasks in the curriculum are sequentially trained by reusing previous policies, and we finally obtain the target policy $\mu^{\Diamond}$ dedicated to the target task $\mathcal{T}^{\Diamond}$. Finally, Algorithm 5 summarizes the SubMas-RL algorithm, including curriculum generation (Line 1) and knowledge transfer (Lines 4–8).

4. Empirical analysis

In this section, we apply the SubMas-RL algorithm to the UAV maneuver scenario and evaluate the performance. The experiments are conducted in an air confrontation simulation platform, XSimStudio, developed by Huaru Technology.[1] XSimStudio has undergone a high-fidelity modeling and simulation development process and has supported for Air Intelligence Game (AIG) as a competition platform.[2]

4.1. Experiment environment and setup

The experimental scenario is the UAV maneuver task, as defined in Section 2.3. Instead of learning from scratch, we apply the SubMas-RL algorithm to build a sequenced curriculum where each task has different numbers of missiles (see Fig. 3). Source task 4 is a simple navigation task with no missiles. We design an expert policy for source task 4 to navigate the evader towards the exit. The training starts from source task 3 by leveraging the expert policy. The state spaces of all tasks are shown in Table 1.

To enhance the policy's robustness, we randomly initialize the positions of both the evader and the interceptor in each episode of the training process. Two metrics, episode return and success rate, are employed to evaluate the performance of our approach. Episode return is the cumulative rewards within one episode, and success rate measures the probability of achieving a successful penetration. The hyperparameters are provided in Table 2; other hyperparameters follow

[1] http://www.huaru.com.cn/product
[2] http://cicc-aig.c2.org.cn/en/index.html

Table 1
State space of source and target tasks.

Task	State space
Target task	$[x, y, \psi, l, \theta_{\text{AA}}, \theta_{\text{ATA}}, \phi_{\text{Exit}}, r_{\text{m}}^1, \phi_{\text{m}}^1, \ldots, r_{\text{m}}^4, \phi_{\text{m}}^4]$
Source task 1	$[x, y, \psi, l, \theta_{\text{AA}}, \theta_{\text{ATA}}, \phi_{\text{Exit}}, r_{\text{m}}^1, \phi_{\text{m}}^1, \ldots, r_{\text{m}}^3, \phi_{\text{m}}^3]$
Source task 2	$[x, y, \psi, l, \theta_{\text{AA}}, \theta_{\text{ATA}}, \phi_{\text{Exit}}, r_{\text{m}}^1, \phi_{\text{m}}^1, r_{\text{m}}^2, \phi_{\text{m}}^2]$
Source task 3	$[x, y, \psi, l, \theta_{\text{AA}}, \theta_{\text{ATA}}, \phi_{\text{Exit}}, r_{\text{m}}^1, \phi_{\text{m}}^1]$
Source task 4	$[x, y, \psi, l, \theta_{\text{AA}}, \theta_{\text{ATA}}, \phi_{\text{Exit}}]$

Table 2
Hyperparameters in SubMas-RL.

Hyperparameter	Value	Note
Network architecture	$[256, 256]$	
Replay buffer size	10^6	$\lvert D \rvert$
Number of epochs	300	
Steps per epoch	10^4	
Learning rate	3e−4	
Batch size	128	
ϵ-start	0.8	ϵ_0
ϵ-clip	0.01	δ_ϵ
α-clip	0.01	δ_α

Table 3
Ablation experiments.

Algorithms	Pd	Pr	α	ϵ
SubMas-RL	√	√	Adaptive	Adaptive
w/oPd	×	√	/	Adaptive
w/oPr	√	×	Adaptive	/
DecPd	√	√	Decayed	Adaptive
DecPr	√	√	Adaptive	Decayed

• Pd: Policy distillation; Pr: Policy reuse.

the default setting in OpenAI SpinningUp.[3] It takes about 30 h to train one task using a single machine (AMD Ryzen Threadripper PRO 3975WX CPU 3.5 GHz with 32 cores, two NVIDIA GeForce RTX 3090 24G GPUs, 128 GB RAM). All experiments are performed over 10 runs with different random seeds.

4.2. Ablation study

We investigate each component of the proposed SubMas-RL algorithm individually to answer the following questions: (i) Does the policy distillation promote knowledge transfer? (ii) Does the policy reuse enhance the sample efficiency? and (iii) How do the adaptive weights α and ϵ affect the performance? We design four ablation algorithms (listed in Table 3): SubMas-RL without policy distillation (w/oPd), SubMas-RL without policy reuse (w/oPr), SubMas-RL with decayed weight α in policy distillation (DecPd), and SubMas-RL with decayed weight ϵ in policy reuse (DecPr). The results of ablation experiments are presented in Fig. 8. Next, we present the main findings.

Policy distillation speeds up the knowledge transfer. The ablation algorithm w/oPd disentangles the role the auxiliary reward plays in knowledge transfer. In Fig. 8, we notice that both methods can realize the same jump-start, indicating that policy distillation has no impact on the jump-start. However, the comparison shows that the w/oPd have a larger standard deviation and a lower asymptotic performance than SubMas-RL. In addition, the w/oPd shows a slight decrease following an initial increase (see the red curve in Fig. 8(a)). We infer that a large jump-start resulting from the high success rate reduces the strength of policy reuse, and the target policy dominates the exploration. However, if policy distillation is not employed, the target policy may learn slowly without the auxiliary reward and fails to adapt to the new task quickly. As for the SubMas-RL algorithm, the agent can utilize the auxiliary reward to rapidly acquire the knowledge encoded in the guide policy. Thus, policy distillation provides dense feedback signals for the agent, accelerating the knowledge transfer.

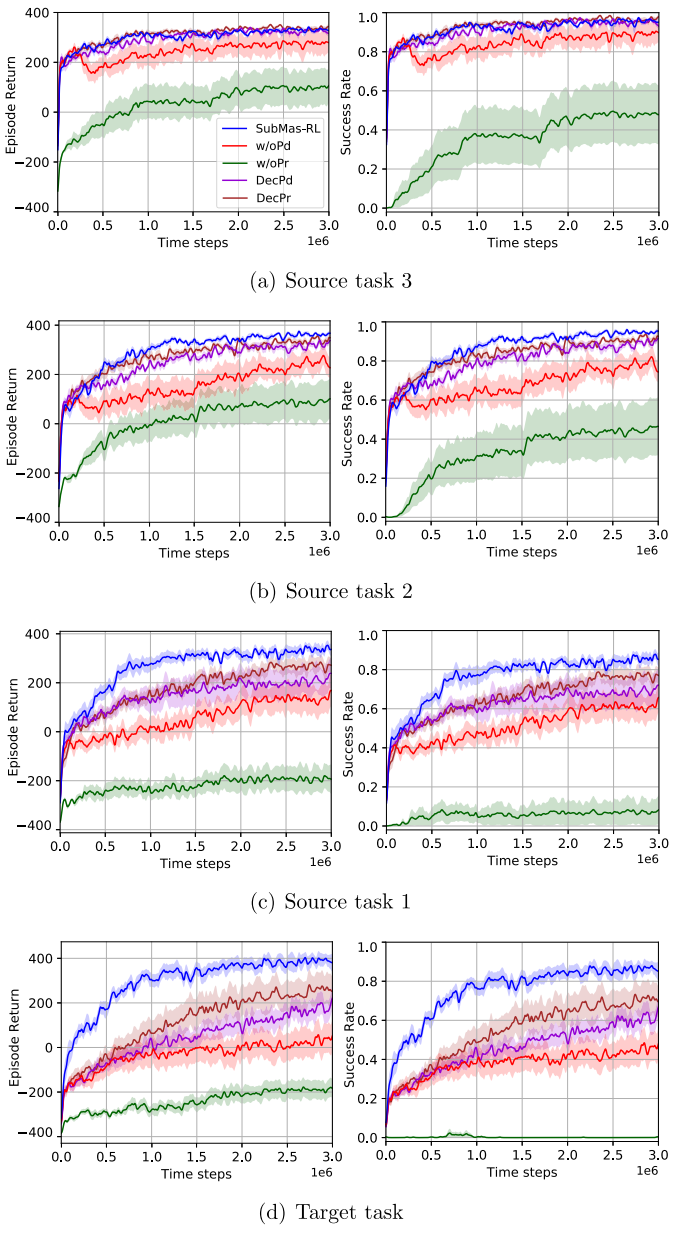


Fig. 8. Ablation study showing episode return (left) and success rate (right) curves of SubMas-RL, w/oPd, w/oPr, DecPd, and DecPr over 10 seeds. The solid lines are the average of all runs, and the shaded area is the standard deviation.

Policy reuse enhances rapid adaptability for a new task. The ablation algorithm w/oPr independently collects samples with the target policy, and the proposed SubMas-RL algorithm samples using the behavior policy (18). Fig. 8 demonstrates that w/oPr suffers from a cold start and is unstable during the training. In contrast, SubMas-RL discovers better trajectories with fewer iterations, resulting in a big jump-start and better asymptotic performance. Obviously, the policy reuse strategy plays a significant role in avoiding the cold start. Although a well-trained guide policy cannot directly achieve optimal performance in the new task, the shared knowledge involved in the guide policy is conducive to sampling near the optimal trajectory. The ϵ-greedy strategy in policy reuse facilitates the balance between utilization and exploration. That is, the agent concentrates on exploring better trajectories after achieving a big jump-start. We also notice that

[3] https://github.com/openai/spinningup/tree/master/spinup/algos/tf1

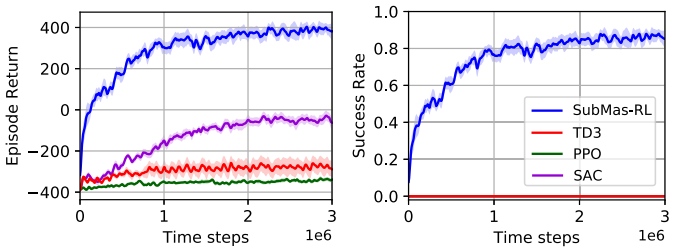


Fig. 9. Comparison study in target task showing episode return (left) and success rate (right) of SubMas-RL, TD3, PPO, and SAC over 10 seeds. The solid lines are the average of all runs, and the shaded area is the standard deviation.

w/oPr achieves an average 50% success rate in source task 3, while the performance in the target task is very poor. In summary, policy reuse significantly enhances rapid adaptability for a new task, especially for more challenging tasks.

The adaptive weights improve the convergence rate. The weights α and ϵ determine the strength of policy distillation and policy reuse, such that we can dynamically control the strength of knowledge transfer. The proposed SubMas-RL algorithm adjusts weights adaptively based on success rates. In ablation, we apply a linear decay to weights α and ϵ (DecPd and DecPr). In Fig. 8(a), we notice that both DecPd, DecPr and SubMas-RL can achieve the same jump-start and asymptotic performance in source task 3. However, in the following tasks (Fig. 8(b)–(d)), DecPd and DecPr perform slightly worse than SubMas-RL due to slower linear decay compared to adaptive weights. It demonstrates that the adaptive weights speed up the convergence.

4.3. Comparison with baselines

In this section, we compare the proposed SubMas-RL algorithm with three baselines: TD3 (Fujimoto et al., 2018), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018). These baselines are employed to learn the target task from scratch. In comparison, SubMas-RL leverages the guide policy generated from the curriculum. The comparative experiments aim to show how the sequenced curriculum benefits the training.

Fig. 9 presents results of comparative experiments, where the SubMas-RL curves are the same as in Fig. 8(d). We observe that SubMas-RL is the only method that has a high success rate. TD3, PPO, and SAC fail to learn the maneuver strategy, and the success rate holds zero during the training. The episode return curves in Fig. 9 show that SAC outperforms TD3 and PPO, which get stuck in local optima. We infer that the better exploration ability of SAC plays an important role in this long-horizon and sparse reward task. However, exploration becomes more challenging with 4 concurrent subtasks, making it hard to devise a successful trajectory without external guidance. SubMas-RL achieves a significant jump-start and better asymptotic performance through the benefits of sequenced curriculum and knowledge transfer.

Then, we evaluate the final policy's performance across all ablation and baseline algorithms. We select the best-performing seed from 10 seeds according to average episode return. Each trained policy executes 1000 episodes on the target task. Table 4 presents a comparison of overall performance between ablations and baselines.

Table 4 shows that SubMas-RL achieves the best performance with an average return of 466.34 and a success rate of 94.8%. Overall, algorithms trained with curriculum learning (SubMas-RL, w/oPd, w/oPr, DecPd, DecPr) achieved better results than those without curriculum learning (TD3, PPO, SAC) on average. It suggests that curriculum learning can help tackle complex tasks beyond the capabilities of standard RL algorithms. However, an interesting exception is that w/oPr shows worse performance compared to TD3 and SAC. Next, an in-depth analysis is undertaken to investigate the underlying mechanisms of all comparable algorithms.

Table 4
Comparison of overall performance between ablations and baselines.

Algorithms	Episode return	Success rate
SubMas-RL	**466.34**	**94.80%**
w/oPd	340.65	80.80%
w/oPr	−59.98	0.00%
DecPd	373.49	85.40%
DecPr	390.16	86.20%
TD3	64.61	0.00%
PPO	−310.70	0.00%
SAC	77.17	0.00%

DecPd and DecPr outperform all comparative algorithms since they preserve critical policy distillation and policy reuse modules, which facilitate efficient knowledge transfer. However, DecPd and DecPr exhibit inferior performance compared to SubMas-RL. This difference can be attributed to the use of different weight types: adaptive weights in SubMas-RL and linear decayed weights in DecPd and DecPr. In contrast to adaptive weights that decline rapidly after initial progress, the linear decayed weights keep larger during the initial training stages. However, larger weights in DecPd and DecPr may enhance the strength of knowledge transfer, thus hindering exploration and adaptation to new tasks.

The w/oPd approach yields an average return of 340.65 and a success rate of 80.80%. In contrast to SubMas-RL, which uses policy distillation to produce auxiliary rewards, w/oPd solely employs policy reuse to improve samples. Due to the absence of dense rewards, it takes a longer training period to transfer knowledge and results in lower overall performance. This finding aligns with the expectation that slower knowledge transfer increases the training period required to attain comparable performance.

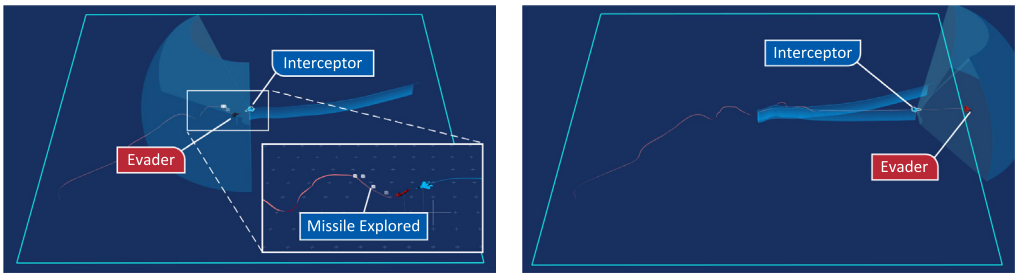
The performance of w/oPr is significantly worse than even TD3 and SAC algorithms, which do not utilize curriculum. This finding raises doubts about the previous finding that the curriculum can promote training. The underlying issue with the poor performance of w/oPr is the inefficient knowledge transfer, which even leads to negative transfer (Wang et al., 2019). Compared with SubMas-RL, which employs policy reuse for efficient sampling, w/oPr can only transfer knowledge slowly through auxiliary rewards. As depicted in Fig. 8, w/oPr achieves only a success rate of 50% in source task 3. As such, incorrect auxiliary rewards are used to train subsequent tasks, which can mislead the training. Despite the curriculum providing guidance for training, efficient knowledge transfer is also crucial for effective learning.

TD3, PPO, and SAC algorithms exhibit improvements in average return during training, but they fail to achieve a positive success rate in the target task due to sparse reward signals and long-horizon sequences. It presents significant challenges for standard RL algorithms to explore and exploit effectively. SAC outperforms the other two algorithms as it incorporates an entropy term into its optimization objective to enhance exploration and avoid local optima. The difference between TD3 and PPO lies in the fact that TD3 employs a deterministic policy while PPO adopts a stochastic policy. Therefore, the potential reason why TD3 outperforms PPO in this task is that the considered UAV maneuver task contains a low degree of stochasticity.

4.4. Comparison with expert policy

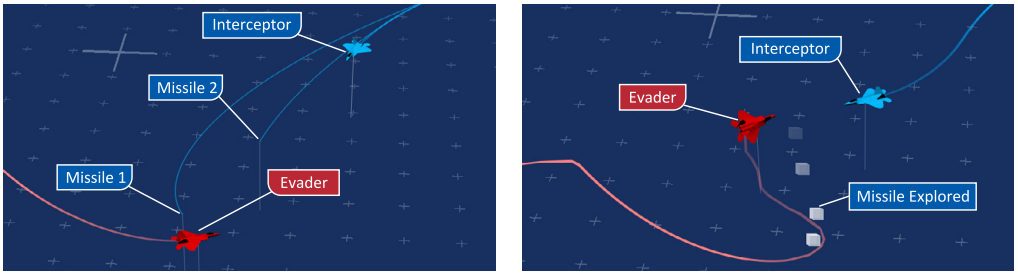
We visualize the policies trained by SubMas-RL, thus understanding what SubMas-RL has learned to achieve the performance. Then, four trained policies are tested in source and target tasks, respectively. The results rendered in XSimStudio are shown in the attached video.

Fig. 10 depicts two screenshots in XSimStudio, showing the performance of SubMas-RL policy in the target task. In the beginning, the evader and interceptor are randomly initialized on two sides. In Fig. 10(a), we observe that the evader maneuvers upwards to occupy a superior position at first rather than directly towards the exit. When



(a) Maneuver and evade four missiles (b) Reach the exit successfully

Fig. 10. Two screenshots of SubMas-RL policy in the target task. (a) The evader has learned to maneuver and evade four missiles continuously. (b) After evading missiles, the evader reach the exit successfully.



(a) Keep the direction vertical to the LOS (b) Successful evading all missiles

Fig. 11. Two screenshots of the expert policy in the target task. (a) The evader always attempts to keep its direction vertical to the missile. (b) The evader successfully evades all missiles.

entering the radar coverage of the interceptor, the evader begins to maneuver and adjust its attitude, even if the interceptor has not yet launched a missile. When the interceptor launches missiles, the evader turns to keep its speed direction (almost) vertical to the LOS of the missiles. This strategy can maximize the energy consumption of missiles, which is similar to defensive dragging (Sun et al., 2021), but with no fixed pattern. After approaching at a certain distance, the evader turns to the missile and flies towards it. When the two sides are very close, the evader violently maneuvers with a large roll angle to evade the missile attack. As shown in Fig. 10(b), after dodging missiles, the evader navigates to the destination successfully.

To figure out the similarity and difference between the SubMas-RL policy and regular maneuvers, we design an expert policy based on the defensive dragging strategy, which is employed by many teams in AIG 2021. Defensive dragging means that the UAV horizontally turns $\pm 90°$ for breaking away from the enemy missile, i.e., keeping its speed direction vertical to the LOS of the missile (Sun et al., 2021). Fig. 11 depicts two screenshots in XSimStudio, showing the performance of the expert policy in the target task. Once the interceptor launches missiles, the evader only takes the closest missile into account and performs defensive dragging. Compared Fig. 11 with Fig. 10, we can observe that the trajectory of defensive dragging is steeper and more violent than the SubMas-RL.

To compare the inherent features of two policies, we visualize the azimuth angle of missiles (ϕ_{m}) in Fig. 12. The angle ϕ_{m} reflects the relative geometry between the evader and missiles. In Fig. 12, curves with different colors refer to different missiles. We can observe that curves of two policies show almost the same trend in the relative geometry. It demonstrates that SubMas-RL has learned a human-relevant behavior without introducing shaping rewards regarding evading missiles. We can also observe in Fig. 12(b) that the defensive dragging strategy always keeps $\pm 90°$ with the closest missile, regardless of other missiles. The curve changed drastically before the missile explosion because the missile missed the UAV and flew from one side to the other. In comparison, SubMas-RL also attempts to keep $\pm 90°$ before the missile explosion, but it prefers to maneuver when approaching the missile.

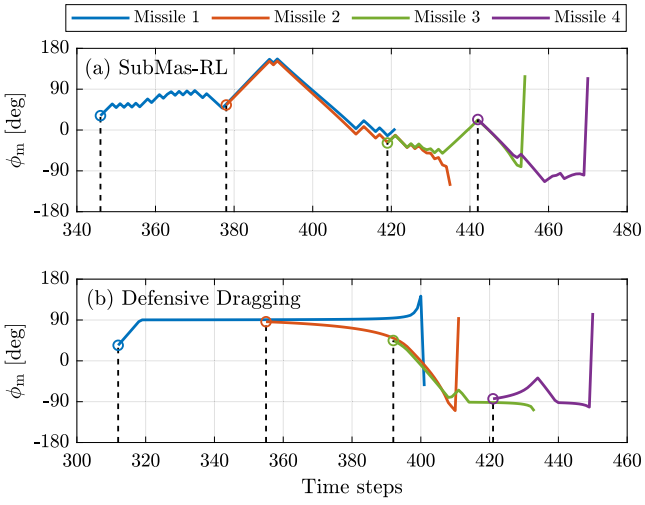


Fig. 12. Curves of the azimuth angle of missiles during evading missiles.

In addition, it is interesting to note the differences between these two policies. Once a new missile is launched, SubMas-RL reacts immediately, and the curves show rapid changes, while defensive dragging only considers the closest missile. Finding the strategy of evading multiple concurrent missiles is the main difference between SubMas-RL and defensive dragging. It is exciting that the SubMas-RL has learned a new maneuver strategy that is similar to the defensive dragging, while obtaining a new strategy regarding multiple concurrent missiles.

Finally, we compare the overall performance of SubMas-RL and expert policy. Each policy is run on the source and target tasks for 1000 episodes, and we evaluate the average episode return and success rate. Fig. 13 presents the performance comparison on SubMas-RL and expert policy over source and target tasks. Taken as a whole, SubMas-RL surpasses the expert policy overall, while the performance of both

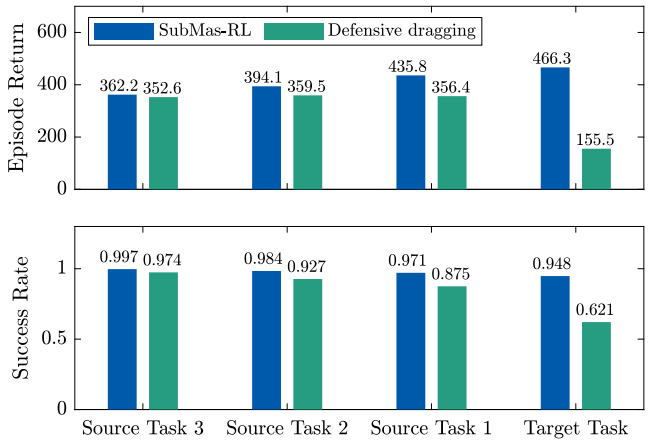


Fig. 13. Episode return and success rate of SubMas-RL (blue) and expert policy (green).

policies decreases as the task becomes more difficult. We can observe that the expert policy suffers from a large decrease in the target task, with only a 62.1% success rate. Multiple missiles and random environments are the major challenges for regular maneuvers. Due to the fixed maneuver strategy, the expert policy fails to generalize its performance in different situations with random initialization. In contrast, the SUBMAS-RL algorithm utilizes randomized training environments and exploratory to enhance the generalization and robustness of the policy.

5. Discussion

5.1. Results analysis

The results of our study suggest that progressively increasing the number of subtasks can effectively induce an agent to master long-horizon, sparse-reward tasks that are beyond the capabilities of standard RL algorithms. A similar work is Dynamic Multi-agent Curriculum Learning (DyMA-CL) (Wang et al., 2020), which masters a large-scale multi-agent scenario by starting from a small size of agents to more agents. Our findings are consistent with those of DyMA-CL, revealing that progressively increasing the difficulty of tasks can significantly enhance the final performance for complex problems.

However, we also observed that in some cases, the curriculum fails to improve performance or even worsens it compared to learning from scratch. The underlying reason for this phenomenon lies in negative transfer, where "bad" guidance can hurt the learning process of the student. In other domains such as vision, previous research has demonstrated that negative transfer is a frequent occurrence but can be overcome (Wang et al., 2019). In the context of reinforcement learning, the negative transfer can be addressed by introducing specific conditions, but it requires enough samples (Zhan et al., 2016). An important contribution of our approach is the incorporation of an explicit transfer condition to avoid negative transfer, which enables efficient knowledge transfer through policy distillation and reuse techniques.

Our results on the UAV maneuver task demonstrate that the proposed SubMas-RL can lead to human-relevant behavior, such as defensive dragging observed in our experiments. More interestingly, meaningful behaviors related to multiple concurrent missiles emerge naturally from the interaction processes, without any need for reward engineering. Our findings are consistent with those reported in similar domains such as air combat, as demonstrated by the results of Sun et al. (2021), which highlight the potential of autocurricula for complex behavior emergence in air combat tactics. Additionally, Baker et al. (2019) demonstrated that autonomous emergence of tool-use behaviors can also be achieved through multi-agent autocurricula. Overall, our results further support the use of curriculum learning for solving complex problems in a natural and autonomous way.

5.2. Limitations

Firstly, it is worth pointing out that SubMas-RL operates under the assumption that the target task can be decomposed into a final goal and multiple subtasks. Unlike existing goal-conditioned hierarchical RL approaches, SubMas-RL does not require subtasks to belong to different temporal scales, but it does require them to be ordered into a sequence according to the difficulty. This condition restricts the practical applicability of SubMas-RL to specific domains.

Secondly, the proposed SubMas-RL only supports deterministic policy gradient algorithms, since the policy distillation term is measured by the distance between specific actions. Nonetheless, stochastic policy gradient algorithms are sometimes preferred to enhance policy exploration. To enable the use of stochastic policy gradient algorithms with SubMas-RL, the policy distillation term should be replaced by the KL divergence between the guide and behavior policies.

Thirdly, our study is subject to potential constraints derived from the designed scenario. A potential limitation of the scenario design is assuming default missile launch intervals in the simulation environment. Our study could benefit from introducing more randomness among opponents in the designed scenario, as training against random opponents has been shown to produce more robust policies (Vinitsky et al., 2020; Wang and Zou, 2021; Xie et al., 2022). Another limitation is that the number of missile threats and the launching platforms in real-world scenarios may vary significantly, potentially limiting the scalability of the proposed solution.

Finally, the constrained training scenario may result in solution outliers and chances of failure. Due to electromagnetic interference and sensor accuracy issues, the agent may fail to obtain accurate observation information about missiles. Without perfect observation, the trained policy cannot generate effective decision-making outcomes. In addition, collisions among UAVs may occur due to spatial and temporal limitations, which might cause damage or even disable the UAVs. Future studies may incorporate more disturbances that may arise in practice to strengthen the practicality of the approach.

6. Related work

Many different paradigms have been proposed to deal with the sample complexity challenges in applying RL to long-horizon tasks with sparse rewards. Transfer learning allows the RL agent to leverage the knowledge from source tasks to benefit the learning in the target task (Taylor and Stone, 2009). Transfer learning is generally employed in curriculum learning to extract and pass on reusable knowledge acquired in one task to the next. Still, it ignores how the source task is generated (Zhu et al., 2020). Our method could easily combine other transfer techniques, such as learning from demonstrations that collect samples from different source tasks and reuse them through inter-task mappings (Nair et al., 2018).

Multi-task learning co-learns a group of different tasks to optimize the performance over all tasks simultaneously (Parisotto et al., 2015). For example, the Distral algorithm trained a central student policy along with several teachers learned in different source domains (Teh et al., 2017). These methods assume that all tasks can be experienced simultaneously and focus on generalized performance (Hessel et al., 2019). In contrast, we focus on rapid adaptation and asymptotic performance on the target task, and source tasks only serve as auxiliary tasks. Continual/Lifelong learning, as an online version of multi-task learning, concentrates on defying catastrophic forgetting and learning new skills without forgetting previous knowledge (Naqushbandi and John, 2022). Compared to these works where tasks are predefined and fixed (Abel et al., 2018), the characteristic of our method is that the order of tasks is well-designed to maximize performance on the target task.

Another related paradigm, meta-reinforcement learning, leverages the task-solving skills trained on a variety of tasks to quickly adapt to a new similar task within a few iterations (Finn et al., 2017; Gupta et al.,

2018; Stadie et al., 2018). In fact, source policies in our method can be regarded as the meta-knowledge in meta-RL to optimize adaptability in the target task (Hu et al., 2021). In essence, meta-RL aims to train for adaptability, whereas our work dedicates to a specific final task. One possible combination to generalize our method with meta-RL is to automatically infer subtask-related information by a latent context network (Sæmundsson et al., 2018) rather than hard-coding inter-task relationships in a domain-specific way.

Furthermore, the challenges of long-horizon tasks and sparse rewards become worse in multi-agent settings. Many multi-agent reinforcement learning (MARL) methods have shown promising results in various tasks (Lowe et al., 2017; Rashid et al., 2020; Xia et al., 2022), but they also suffer from high sample complexity and scalability issues, especially when the number of agents is large. The integration of MARL and curriculum learning is a promising direction for tackling complex and multi-agent tasks. Wang et al. (2020) proposes a dynamic curriculum learning approach that gradually increases the number of agents in the environment to improve the learning efficiency of MARL, and shows that it can be combined with transfer learning to further enhance performance. Perhaps a possible direction is to extend our method to the multi-agent setting, which will be further investigated in our future work.

7. Conclusions

In this paper, we propose an efficient algorithm, Subtask-Masked curriculum learning for Reinforcement Learning (SubMas-RL), to address the complex UAV maneuver task with sparse reward settings. We present the concepts of subtask mask and transfer mask for generating a sequenced curriculum by masking subtasks while avoiding negative transfer among the curriculum. We also introduce two transfer mechanisms across different curricula to accelerate the learning process, which is extensively validated by simulations. Experimental results demonstrate that the proposed algorithm achieves a high success rate of 94.8% in the UAV maneuver task with four concurrent missiles, where direct implementation of standard RL algorithms often fails. Furthermore, SubMas-RL obtains a human-relevant strategy, and meaningful behaviors related to multiple concurrent missiles emerge naturally from the interaction processes. The main limitations of our study are the assumption on subtasks and potential constraints derived from the designed scenario.

Possible directions of future research include: introducing subtask metrics to realize subtask sequencing and curriculum generation automatically, combining the proposed approach with stochastic policy gradient algorithms to broaden its applicability, building a high-fidelity simulation environment, designing more dynamic scenarios and opponents to further enhance the practicality, and leveraging distributed sampling and diverse adversarial pools to enhance policy robustness.

CRediT authorship contribution statement

Yueqi Hou: Methodology, Software, Writing – original draft, Writing – review & editing, Visualization. **Xiaolong Liang:** Conceptualization, Supervision, Project administration. **Maolong Lv:** Formal analysis, Resources, Methodology. **Qisong Yang:** Methodology, Data curation, Visualization. **Yang Li:** Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, and there is no any commercial or associative interest that represents a conflict of interest in connection with our manuscript entitled "***Subtask-Masked Curriculum Learning for Reinforcement Learning with Application to UAV Maneuver Decision-Making***".

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Young Talent Fund of Association for Science and Technology in Shaanxi, China (grant no. 20220101), and the Young Talent Support Project for Military Science and Technology. The authors would also like to thank Huaru Technology for providing the simulation platform (XSimStudio).

References

Abel, D., 2022. A theory of abstraction in reinforcement learning. arXiv preprint arXiv:2203.00397.

Abel, D., Arumugam, D., Lehnert, L., Littman, M., 2018. State abstractions for lifelong reinforcement learning. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 10–19.

Ayeelyan, J., Lee, G.H., Hsu, H.C., Hsiung, P.A., 2022. Advantage actor-critic for autonomous intersection management. Vehicles 4 (4), 1391–1412.

Bacon, P.L., Harb, J., Precup, D., 2017. The option-critic architecture. In: Proc. Conf. AAAI Artif. Intell., Vol. 31. (1).

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I., 2019. Emergent tool use from multi-agent autocurricula. arXiv preprint arXiv:1909.07528.

Choi, Y.B., Jin, S.H., Kim, K.S., Chung, B.D., 2018. A robust optimization approach for an artillery fire-scheduling problem under uncertain threat. Comput. Ind. Eng. 125, 23–32. http://dx.doi.org/10.1016/j.cie.2018.08.015.

Crespi, V., Galstyan, A., Lerman, K., 2008. Top-down vs bottom-up methodologies in multi-agent system design. Auton. Rob. 24, 303–313.

Eysenbach, B., Gupta, A., Ibarz, J., Levine, S., 2018. Diversity is all you need: Learning skills without a reward function. arXiv preprint arXiv:1802.06070.

Fernández, F., Veloso, M., 2006. Probabilistic policy reuse in a reinforcement learning agent. In: Proc. 15th Int. Joint Conf. Auton. Agents Multiagent Syst.. pp. 720–727.

Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 1126–1135.

Florensa, C., Duan, Y., Abbeel, P., 2017. Stochastic neural networks for hierarchical reinforcement learning. arXiv preprint arXiv:1704.03012.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 1587–1596.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., Levine, S., 2018. Meta-reinforcement learning of structured exploration strategies. Adv. Neural Inf. Process. Syst. 31.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 1861–1870.

He, Y., Wang, Y., Lin, Q., Li, J., 2022. Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks. IEEE Trans. Veh. Technol. 71 (4), 3495–3506. http://dx.doi.org/10.1109/TVT.2022.3146439.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., van Hasselt, H., 2019. Multi-task deep reinforcement learning with popart, vol. 33. In: Proc. Conf. AAAI Artif. Intell.. (01), pp. 3796–3803.

Hou, Y., Liang, X., Zhang, J., Lv, M., Yang, A., 2023. Hierarchical decision-making framework for multiple UCAVs autonomous confrontation. IEEE Trans. Veh. Technol. 1–16. http://dx.doi.org/10.1109/TVT.2023.3285223.

Hu, Y., Gao, Y., An, B., 2015. Accelerating multiagent reinforcement learning by equilibrium transfer. IEEE Trans. Cybern. 45 (7), 1289–1302. http://dx.doi.org/10.1109/TCYB.2014.2349152.

Hu, H., Huang, G., Li, X., Song, S., 2021. Meta-reinforcement learning with dynamic adaptiveness distillation. IEEE Trans. Neural Netw. Learn. Syst. http://dx.doi.org/10.1109/TNNLS.2021.3105407.

Hu, H., Huang, G., Li, X., Song, S., 2023. Meta-reinforcement learning with dynamic adaptiveness distillation. IEEE Trans. Neural Netw. Learn. Syst. 34 (3), 1454–1464. http://dx.doi.org/10.1109/TNNLS.2021.3105407.

Hu, D., Yang, R., Zhang, Y., Yue, L., Yan, M., Zuo, J., Zhao, X., 2022. Aerial combat maneuvering policy learning based on confrontation demonstrations and dynamic quality replay. Eng. Appl. Artif. Intel. 111, 104767. http://dx.doi.org/10.1016/j.engappai.2022.104767.

Ivanovic, B., Harrison, J., Sharma, A., Chen, M., Pavone, M., 2019. Barc: Backward reachability curriculum for robotic reinforcement learning. In: Proc. IEEE Int. Conf. Robot Autom.. IEEE, pp. 15–21.

Jia, Y., Qu, L., Li, X., 2022. A double-layer coding model with a rotation-based particle swarm algorithm for unmanned combat aerial vehicle path planning. Eng. Appl. Artif. Intel. 116, 105410. http://dx.doi.org/10.1016/j.engappai.2022.105410.

Levine, S., Finn, C., Darrell, T., Abbeel, P., 2016. End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. 17 (1), 1334–1373.

Levy, A., Konidaris, G., Platt, R., Saenko, K., 2017. Learning multi-level hierarchies with hindsight. arXiv preprint arXiv:1712.00948.

Li, G., Ji, Z., Li, S., Luo, X., Qu, X., 2023. Driver behavioral cloning for route following in autonomous vehicles using task knowledge distillation. IEEE Trans. Intell. Veh. 8 (2), 1025–1033. http://dx.doi.org/10.1109/TIV.2022.3198678.

Liang, Z., Liang, W., Wang, Z., Ma, X., Liu, L., Zhu, Z., 2022. Multiobjective evolutionary multitasking with two-stage adaptive knowledge transfer based on population distribution. IEEE Trans. Syst. Man Cybern.: Syst. 52 (7), 4457–4469. http://dx.doi.org/10.1109/TSMC.2021.3096220.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Adv. Neural Inf. Process. Syst. 30.

McGrew, J.S., How, J.P., Williams, B., Roy, N., 2010. Air-combat strategy using approximate dynamic programming. J. Guid. Control Dyn. 33 (5), 1641–1654.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Nagpal, R., Krishnan, A.U., Yu, H., 2020. Reward engineering for object pick and place training. arXiv preprint arXiv:2001.03792.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., Abbeel, P., 2018. Overcoming exploration in reinforcement learning with demonstrations. In: Proc. IEEE Int. Conf. Robot Autom.. IEEE, pp. 6292–6299.

Naqushbandi, F.S., John, A., 2022. Sequence of actions recognition using continual learning. In: 2022 Second International Conference on Artificial Intelligence and Smart Energy. ICAIS, pp. 858–863. http://dx.doi.org/10.1109/ICAIS53314.2022.9742866.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P., 2020. Curriculum learning for reinforcement learning domains: A framework and survey. J. Mach. Learn. Res. 21.

Narvekar, S., Sinapov, J., Leonetti, M., Stone, P., 2016. Source task creation for curriculum learning. In: Proc. 15th Int. Conf. Auton. Agents Multiagent Syst.. pp. 566–574.

Parisotto, E., Ba, J.L., Salakhutdinov, R., 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. arXiv preprint arXiv:1511.06342.

Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S., 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. J. Mach. Learn. Res. 21 (1), 7234–7284.

Rosman, B., Hawasly, M., Ramamoorthy, S., 2016. Bayesian policy reuse. Mach. Learn. 104, 99–127.

Rusu, A.A., Colmenarejo, S.G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., Hadsell, R., 2015. Policy distillation. arXiv preprint arXiv:1511.06295.

Sæmundsson, S., Hofmann, K., Deisenroth, M.P., 2018. Meta reinforcement learning with latent variable gaussian processes. arXiv preprint arXiv:1803.07551.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Silva, F.L.D., Costa, A.H.R., 2018. Object-oriented curriculum generation for reinforcement learning. In: Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.. pp. 1026–1034.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms. In: Proc. Int. Conf. Mach. Learn.. PMLR, pp. 387–395.

Sohn, S., Oh, J., Lee, H., 2018. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. Adv. Neural Inf. Process. Syst. 31.

Stadie, B.C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., Sutskever, I., 2018. Some considerations on learning to explore via meta-reinforcement learning. arXiv preprint arXiv:1803.01118.

Sun, X., Cai, C., Yang, J., Shen, X., 2015. Route evaluation for unmanned aerial vehicle based on type-2 fuzzy sets. Eng. Appl. Artif. Intel. 39, 132–145. http://dx.doi.org/10.1016/j.engappai.2014.11.008.

Sun, Z., Piao, H., Yang, Z., Zhao, Y., Zhan, G., Zhou, D., Meng, G., Chen, H., Chen, X., Qu, B., et al., 2021. Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play. Eng. Appl. Artif. Intel. 98, 104112.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., Stone, P., 2017. Automatic curriculum graph generation for reinforcement learning agents. In: Proc. Conf. AAAI Artif. Intell., Vol. 31. (1).

Taylor, M.E., Stone, P., 2007. Cross-domain transfer for reinforcement learning. In: Proc. 24th Int. Conf. Mach. Learn.. pp. 879–886.

Taylor, M.E., Stone, P., 2009. Transfer learning for reinforcement learning domains: A survey.. J. Mach. Learn. Res. 10 (7).

Taylor, M.E., Stone, P., Liu, Y., 2007. Transfer learning via inter-task mappings for temporal difference learning.. J. Mach. Learn. Res. 8 (9).

Teh, Y., Bapst, V., Czarnecki, W.M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., Pascanu, R., 2017. Distral: Robust multitask reinforcement learning. Adv. Neural Inf. Process. Syst. 30.

Vinitsky, E., Du, Y., Parvate, K., Jang, K., Abbeel, P., Bayen, A., 2020. Robust reinforcement learning using adversarial populations. arXiv preprint arXiv:2008.01825.

Wang, Z., Dai, Z., Póczos, B., Carbonell, J., 2019. Characterizing and avoiding negative transfer. In: 2019 IEEE/CVF Conf. Comput. Vision Pattern Recognit.. CVPR, pp. 11285–11294. http://dx.doi.org/10.1109/CVPR.2019.01155.

Wang, H., Tao, C., Qi, J., Xiao, R., Li, H., 2022. Avoiding negative transfer for semantic segmentation of remote sensing images. IEEE Trans. Geosci. Remote Sens. 60, 1–15. http://dx.doi.org/10.1109/TGRS.2022.3201688.

Wang, W., Yang, T., Liu, Y., Hao, J., Hao, X., Hu, Y., Chen, Y., Fan, C., Gao, Y., 2020. From few to more: Large-scale dynamic multiagent curriculum learning. In: Proc. Conf. AAAI Artif. Intell., Vol. 34. (05), pp. 7293–7300.

Wang, Y., Zou, S., 2021. Online robust reinforcement learning with model uncertainty. Adv. Neural Inf. Process. Syst. 34, 7193–7206.

Xia, W., Zhu, Y., De Simone, L., Dagiuklas, T., Wong, K.K., Zheng, G., 2022. Multiagent collaborative learning for UAV enabled wireless networks. IEEE J. Sel. Areas Commun. 40 (9), 2630–2642. http://dx.doi.org/10.1109/JSAC.2022.3191329.

Xie, A., Sodhani, S., Finn, C., Pineau, J., Zhang, A., 2022. Robust policy learning over multiple uncertainty sets. In: Int. Conf. Mach. Learn.. PMLR, pp. 24414–24429.

Xu, S., Zhang, X., Li, C., Wang, D., Yang, L., 2022. Deep reinforcement learning approach for joint trajectory design in multi-UAV IoT networks. IEEE Trans. Veh. Technol. 71 (3), 3389–3394. http://dx.doi.org/10.1109/TVT.2022.3144277.

Yang, Q., Simão, T.D., Jansen, N., Tindemans, S.H., Spaan, M.T., 2022. Training and transferring safe policies in reinforcement learning. In: Adaptive Learning Agents Workshop at AAMAS.

Yang, Q., Spaan, M.T., 2023. Cem: constrained entropy maximization for task-agnostic safe exploration. In: Proc. Conf. AAAI Artif. Intell., Vol. 37. (9), pp. 10798–10806. http://dx.doi.org/10.1609/aaai.v37i9.26281.

Yang, C.D., Yang, C.C., 1996. Analytical solution of 3D true proportional navigation. IEEE Trans. Aerosp. Electron. Syst. 32 (4), 1509–1522. http://dx.doi.org/10.1109/7.543873.

Yang, Z., Zhou, D., Piao, H., Zhang, K., Kong, W., Pan, Q., 2020. Evasive maneuver strategy for UCAV in beyond-visual-range air combat based on hierarchical multi-objective evolutionary algorithm. IEEE Access 8, 46605–46623. http://dx.doi.org/10.1109/ACCESS.2020.2978883.

Yomchinda, T., 2015. A study of autonomous evasive planar-maneuver against proportional-navigation guidance missiles for unmanned aircraft. In: 2015 Asian Conf. Def. Technol.. ACDT, pp. 210–214. http://dx.doi.org/10.1109/ACDT.2015.7111613.

Zhan, Y., Ammar, H.B., et al., 2016. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. arXiv preprint arXiv:1604.03986.

Zhang, T., Guo, S., Tan, T., Hu, X., Chen, F., 2020. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. Adv. Neural Inf. Process. Syst. 33, 21579–21590.

Zhang, T., Wang, X., Liang, B., Yuan, B., 2022. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. IEEE Trans. Neural Netw. Learn. Syst. 1–15. http://dx.doi.org/10.1109/TNNLS.2022.3162241.

Zhu, Z., Lin, K., Zhou, J., 2020. Transfer learning in deep reinforcement learning: A survey. arXiv preprint arXiv:2009.07888.