Turkish Journal of Science & Technology
18(1), 243-259, 2023

Research Paper
https://doi.org/10.55525/tjst.1219845

A Hierarchical Reinforcement Learning Framework for UAV Path Planning in Tactical Environments

Mahmut Nedim ALPDEMİR[1*]
[1] BİLGEM, TÜBİTAK, Ankara, Türkiye
[*1] nedim.alpdemir@tubitak.gov.tr

(Geliş/Received: 16/12/2022; Kabul/Accepted: 21/03/2023)

Abstract: Tactical UAV path planning under radar threat using reinforcement learning involves particular challenges ranging from modeling related difficulties to sparse feedback problem. Learning goal-directed behavior with sparse feedback from complex environments is a fundamental challenge for reinforcement learning algorithms. In this paper we extend our previous work in this area to provide a solution to the problem setting stated above, using Hierarchical Reinforcement Learning (HRL) in a novel way that involves a meta controller for higher level goal assignment and a controller that determines the lower-level actions of the agent. Our meta controller is based on a regression model trained using a state transition scheme that defines the evolution of goal designation, whereas our lower-level controller is based on a Deep Q Network (DQN) and is trained via reinforcement learning iterations. This two-layer framework ensures that an optimal plan for a complex path, organized as multiple goals, is achieved gradually, through piecewise assignment of sub-goals, and thus as a result of a staged, efficient and rigorous procedure.

Keywords: Hierarchical reinforcement learning, tactical UAV path planning, machine learning, path optimization.

Taktik Ortamlarda IHA Güzergahı Planlama Amaçlı Hiyerarşik Pekiştirmeli Öğrenmeye Dayalı bir Çerçeve

Öz: Pekiştirmeli öğrenme ile radar tehditi altındaki İHA'ların güzergah planlamasının yapılması, modelleme ile ilintili zorluklardan başlayıp seyrek geri-besleme (sparse feedback) problemine uzanan bir dizi zorluklar içerir. Esasen, amaca yönelik davranışların seyrek geri beslemenin söz konusu olduğu karmaşık ortamlarda öğrenilmesi hususu pekiştirmeli öğrenme algoritmaları açısından temel bir problemdir. Bu makalede, daha önce bu alanda yaptığımız bir çalışmanın genişletilmesi suretiyle, bahse konu bu probleme, hiyerarşik pekiştirmeli öğrenme yöntemine dayalı özgün bir çözüm önerilmektedir. Bu çözüm, aktörlere üst seviye hedeflerin atanmasını yöneten bir meta-denetleyici ile aktörün alt seviye eylemlerini yöneten bir alt-denetleyicinin hiyerarşik bir şekilde çalışması prensibine dayanmaktadır. Çözümümüzdeki meta-denetleyici, yüksek-seviye hedeflerin atanması sürecini tanımlayan bir durum geçiş düzeneği kullanılarak eğitilen bir regresyon modeline, alt-düzey denetleyici ise pekiştirmeli öğrenme ile eğitilen bir DQN'e dayanmaktadır. Bu iki katmanlı çerçeve, birden fazla hedef şeklinde organize edilmiş karmaşık bir güzergah için en iyi planın, hedeflerin aşamalı olarak atanması suretiyle ilerleyen; kademeli, etkin ve sıkı bir yöntem ile tedricen başarılmasını sağlamaktadır.

Anahtar kelimeler: Hiyerarşik pekiştirmeli öğrenme, taktik İHA güzergah planlama, makine öğrenmesi, güzergah optimizasyonu.

1. Introduction

The problem of Tactical UAV (TUAV) path optimization under radar threat includes several constraints which make it different from other path planning problems. Perhaps the most important of these constraints is the requirement to evade the detection and tracking of radars capable of undermining the survivability of the UAV. In [37] we have illustrated the integrated use of RL methods and a transfer learning approach in a single framework to solve the UAV path optimization problems in a tactical environment. As stated in [37] there are a number of peculiarities in comparison to more conventional problem settings, regarding UAV path optimization and the problem of Tactical UAV path planning under hostile radar tracking threat. To recap, these can be summarized as follows:

1. Radar coverage areas exhibit different properties from a penetration point of view, since depending on the performance parameters of the radars, those 3D regions would translate into different threat levels for a potential target. Moreover, those regions may be overlapping as opposed to conventional physical obstacles.
2. In a tactical context, some radars have an associated response period and a missile impact-to-kill time, leading to legitimate penetration periods without impairing the survivability of the target.

* Corresponding author: nedim.alpdemir@tubitak.gov.tr. ORCID Number of author: [1] 0000-0001-6411-1453

3. Target detection is achieved via the interaction of the propagating radar EM waves with the Radar Cross Section (RCS) of the target which is a function of multiple parameters such as the radar operating frequency, the shape of the UAV and more importantly the engagement geometry between the radar and the UAV. The engagement geometry, in particular, implies that multiple angles (such as the bank angle, the aspect angle and elevation angle) would change as a result of the maneuvers performed by the UAV. So, even if the slant range (i.e., direct distance) to the radar is relatively stable, its detectability can still fluctuate dramatically.

4. The operating environments tend to be larger compared to in-door or small-sized urban areas, which implies that the feedback signals leading the agent towards the goal(s) may be delayed. This induces a well-known complication termed as *sparse reward problem*.

In fact, learning a task in a complex environment where sparse-feedback is an issue, is a significant challenge for artificial intelligence as noted in [16]. In such environments, the learning process should either incorporate a structured and effective way of representing the knowledge at multiple levels of spatio-temporal abstractions, or otherwise it should make sure that the reward signals provide sufficient and timely feedback to direct the agent effectively towards the goal. In practice, a variant of the general path planning problem may emerge when there are a set of well-defined sub-tasks during the course of the total mission flight time. The interim tasks can be as simple as passing along a pre-defined fixed location, or they may require more complex logic which may involve a set of constraints that must be satisfied at multiple levels. In reinforcement learning terms this translates into a *hierarchical learning process* guided via a multiple goal structure, leading to the concept of *Hierarchical Reinforcement Learning (HRL)*. As such, HRL inherently caters for sparse reward problem and conveniently accommodates a class of problems involving multiple tasks with a well-defined arrangement.

However, HRL is a specific form of the general problem with non-trivial implications and therefore induces some adjustments to the underlying formal model (i.e., Markov Decision Process (MDP)) as well as to the architecture of the framework providing the solution to the problem. In particular, from a theoretical point of view, the fact that actions of the agent may now be organized with regard to a set of temporal and/or logical abstractions, breaks the Markov Principle which states that the transition from current (present) state (S_t) to the next state (S_{t+1}) is entirely independent of the past (i.e. the previous states). In other words, Markov Property entails that the actions of the agent must be atomic in time and the current state already captures the information of the past states. Relaxation of this principle leads to a variant of MDP known as *Semi-Markov Decision Process (SMDP)* [39], [40]. From an implementation point of view, on the other hand, the framework providing the solution should include a layered architecture where higher level(s) would deal with assignment and learning of more abstract tasks, and lower level(s) would handle optimum action selection for the agent towards achieving a particular (sub)task.

In this paper we extend our previous work reported in [37] to provide a solution to the problem setting stated above, using hierarchical reinforcement learning in a novel way that involves a meta controller for higher level goal assignment and a controller that determines the lower-level actions of the agent. Our meta controller is based on a regression model trained using a state transition scheme that defines the evolution of goal designation, whereas our lower-level controller is based on a Deep Q Network (DQN) [25] and is trained via reinforcement learning iterations. This two-layer framework ensures that an optimal plan for a complex path, organized as multiple goals, is achieved gradually, through piecewise assignment of sub-goals, and thus as a result of a staged, efficient and rigorous procedure.

The rest of the paper is organized as follows: Section 2 provides some background and related work and introduces the details of the environment modeling; Section 3 sketches the theoretical framework of Reinforcement Learning, Section 4 introduces the specific RL formulation of the TUAV path optimization problem, Section 5 presents the experiments, results and discussions, and finally Section 6 provides the concluding remarks.

2. Background And Environment Modeling

TUAV path optimization under radar tracking threat has been explored by a number of researchers in non-RL settings [33, 21, 36, 17, 35, 12, 11]. The methods proposed by these researchers include Nonlinear Trajectory Generation (NTG) algorithm [11], Label Setting Algorithm (LSA) [33], A^* algorithm [21, 35], numerical optimization procedure for a minimax optimal control, with moving average functional [12], Among the non-RL based solutions the one presented in [12] is utilized in our work since it integrates the model of the aircraft, a probabilistic model of a radar, and a behavioral approximation of missile subsystems based on the decision process for launching a SAM and the requirement to maintain tracking during missile flyout. Two of the components, namely radar and RCS models, however, are extended to incorporate pulse integration capability of a radar and to

account for the impact of target fluctuation via the Swerling type, which can have a significant effect for airborne platforms with a certain agility.

To put the path optimization problem in perspective with respect to RL, it is important to note that RL can be thought as a form of simulation-based, approximate dynamic programming, and as stated in [8], it provides a viable solution to such optimization problems. In fact, the ability of RL to tackle control and optimization problems is well-appreciated in the literature [27, 2, 24]. As such, more and more research work are being reported which involve RL based solutions. In a recent survey, AlMahamid and Grolinger [37] provide a systematic review of 159 recent works reported between 2016 and 2021, that use RL as a solution to UAV navigation and path planning problems. We note that none of the covered papers in this comprehensive review proposes a HRL-based solution to tactical UAV path optimization problem under radar threat. Incidentally, use of HRL in UAV path planning problems, in general, has been on the agenda only very recently. One of the quite recent works that resort to HRL as a solution in the UAV path planning context is that of Cheng Y. et.al [44] who aim to model cooperative formation of swarm UAVs. They combine a variant of MAXQ approach and simulated annealing to accomplish grid method-based path planning for multi-UAV collaborative formation. They report quicker astringence, lower volatility, better learning effect, less time consumed and optimized searched route compared to non-hierarchical RL methods. Their problem setting is different to ours, in that their main objective is to achieve collaborative formation of multiple UAVs and that they do not consider evading lethal threats in tactical environments. Another recent work that proposes an HRL-based solution to missile guidance problem with threat-region (i.e., circular no-fly zones) avoidance is that of Bohao et.al. [46], who use an improved version of hierarchical deep deterministic policy gradient (DDPG) algorithm. From a problem-context point of view their objective has some similarities to ours since it involves navigation to a target (although using UAVs) and threat avoidance. However, from a methodological point of view, there are important differences. Their purpose of employing hierarchical learning is to weaken acceleration chattering rather than constructing a temporal abstraction for staged navigation. They achieve this by employing a somewhat shallow hierarchy of two DDPG value networks sharing the same policy network. Being a DDPG based solution, their approach uses a continuous action space, a policy-gradient learning method and an actor-critic architecture in each of the DDPG component. Our solution is based on a strictly layered architecture, uses discrete action space and is based on learning directly the optimum Q-Values (rather than the optimum policy). The last research work we would like to compare is that of Qin et.al. [50] who propose to solve the trajectory design problem of rechargeable UAVs in continuous data collection tasks using a hierarchical DQN (H-DQN) architecture based on SMDP. Their problem setting is significantly different to ours in that higher level tasks (called options in their paper akin to the options formalism of Sutton et.al,[47]) pertain to pre-defined UAV behaviors such as data collection, returning to base for re-charge etc. Whereas in our work, higher level tasks are interim sub-goals inspired by navigational targets. From an algorithmic point of view their method has similarities in particular in the way low level tasks are learned, which involves training a DQN network with experience replay using epsilon greedy exploration. The main methodological difference is that our framework is more akin to Hierarchies of Abstract Machines (HAMs) of Parr [39] and our learning scheme is based on Feudal RL (more on this in Section 3 and Section 4.)

In the following subsections we provide the details of the modeling aspects that are important for understanding the underlying logic of the environment state generation in our framework.

2.1 Modeling of the RL Environment

The modeling related aspects of our RL environment were developed in [37], here we include a concise summary of the mathematical model that underpins the overall response of the environment with respect to the interactions of the agent. For details that cannot be found in this paper we refer the reader to the aforementioned publication.

2.1.1 A Probabilistic Model of Radar for detection, tracking and destruction of a target

From a probabilistic point of view, based on the SNR values and a choice of false alarm rate (P_{fa}) it is possible to compute a *probability of detection* for a combination of pulse integration capability of the radar and the type of the target behavior. [18] states that for a single pulse (no pulse integration) radar and non-fluctuating target the following approximation gives the solution:

$$P_d \approx 0.5 \times erfc\left(\sqrt{-ln(P_{fa}) - \sqrt{SNR + 0.5}}\right) \quad (1)$$

In the context of tactical UAV survivability, target *tracking* is more important than target *detection*, since for a successful engagement, a track lock has to be acquired and maintained on the target for a certain period of time. Therefore, the *probability of track* has to be included in the model. [34] states that given a P_d value and a probabilistic account of the track loss, the probability of track of a radar can be calculated in a recursive way using the following Equation:

$$z[n+1] = (1 - z[n-l])P_d \times (1 - P_m)^l + z[n] \qquad (2)$$
$$P_{track} = 1 - z[k];\ \ n > l, \qquad and\ \ P_m = 1 - P_d$$

where a track loss is set to occur only after l number of consecutive misses in a k number of such observations, subject to $k > l$. However, for an effective and convenient computational model of radar-target engagement, a direct relationship between critical parameters of the engagement is required. [12] proposes a concise formulation where the relationship between the target range, the RCS of the target and the probability of track is established and parameterized by a combination of performance and operational characteristics of the radar.

$$P(R, \sigma) = \frac{1}{1+\left(c_2 \times \frac{R^4}{\sigma}\right)^{c_1}} \qquad (3)$$

The function $P(R, \sigma)$ given above in Equation (3), is a direct function of range (R) and RCS (σ) where c_1 and c_2 are constants and can be determined using the operational parameters and some required performance characteristics of a radar. To account for cases where a number of radars are installed to form a tactical network, the aggregate probability of being tracked by a number of those radars has to be computed. Under the assumption that tracking logic of the participating radars is not shared among them the aggregate probability can be evaluated as follows;

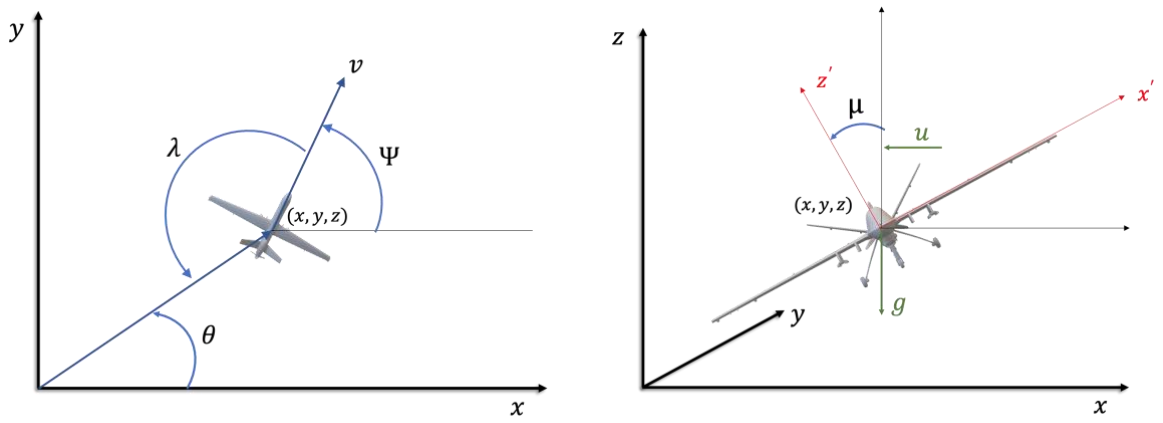
$$P_{tr} = 1 - (1 - P_{tr_1})(1 - P_{tr_2}) \ldots (1 - P_{tr_n}) \qquad (4)$$

Where P_{tr_1} denotes the probability of the UAV being tracked by a first radar, P_{tr_2} denotes the probability of the UAV being tracked by a second radar and so on, and P_{tr} is the probability of the event that the UAV is tracked by an overall integrated air defense network consisting of all radars in a given mission area.

The last element of our probabilistic engagement model, is the probability of the TUAV to be hit by a missile after it has been detected and a track lock is acquired on it. In this respect, two primary factors have to be considered: 1 - system response time, T_r (i.e., the time interval in which the radar system concludes that a missile must be fired and actually fires it), 2 - missile flyout time, T_f (the time interval in which a missile travels to physically hit or explode in proximity to the aircraft). For an aircraft to be destroyed at time t, the radar system must have tracked it during the continuous interval $[t - (T_r + T_f), t]$. If $\Delta T = T_r + T_f$ is defined to be the threat window, then probability of kill is defined as:

$$P_k(t) \lessapprox \frac{1}{\Delta T} \int_{t-\Delta T}^{t} P_t(\tau) d\tau \qquad (5)$$

2.1.2 A Kinematic Bank-to-Turn Model of the UAV



(a) $\lambda, \theta \; and \; \psi$ angles from an orthogonal top view (b) μ is the bank (roll) angle

Figure 1 A Visual representation of UAV model parameters

The UAV is represented using a kinematic model given in [12] and also adopted in [36], which account for the coupling between the RCS and aircraft dynamics. To be more specific, the turn rate of the aircraft is determined by its bank angle which in turn is determined by a steering input represented by u. Hence, the RCS and dynamics of the aircraft are coupled through the aspect and bank angles.

Figure 1 (a) shows an orthogonal top view to indicate the relationship between $\lambda, \theta \; and \; \psi$ angles.

Figure 1 (b) shows a tilted 3D view of a UAV to illustrate the relationship between μ (the bank (roll) angle) $u \; and \; g$. According to this model, the bank-to-turn aircraft is assumed to move in a horizontal plane at a constant altitude governed by the equations:

$$\dot{x} = v \; cos \; \psi, \tag{6}$$
$$\dot{y} = v \; sin \; \psi, \tag{7}$$
$$\dot{\psi} = u/v \tag{8}$$

Where, x and y are Cartesian coordinates of the UAV, ψ is the heading angle, v is the constant speed, u is the input signal which is the acceleration, normal to the flight path vector. Further, to define the position and posture of the aircraft with respect to a given i_{th} radar the following equations are defined [1]:

$$\theta = arctan\left(\frac{y-y_i}{x-x_i}\right), \tag{9}$$
$$\lambda = \theta \; - \psi \; + \; \pi \tag{10}$$
$$\phi = arctan\left(\frac{z}{\sqrt{(x-x_i)^2+(y-y_i)^2}}\right), \tag{11}$$
$$\mu = arctan(u/g) \tag{12}$$
$$R_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + z^2} \tag{13}$$

Where, θ, λ, ϕ, μ are the azimuth, aspect, elevation and bank (or roll) angles, respectively, z is the aircraft altitude, x_i, y_i are the coordinates of the i_{th} radar on the $x-y$ plane (note that $z = 0$ for any radar), g is the acceleration of gravity and R_i is the range of the i_{th} radar to the aircraft. To ensure a certain degree of faithfulness to the actual RCS of the aircraft rather than rely on a simple isotropic RCS, we adopt the optical region approximations using a 3D Elliptical RCS model, which is one of the suggested approximations in the literature [18]. Based on Equations (9), (10), (11) and (12), the RCS of the aircraft can be modeled as a function of the aspect angle λ, the elevation angle ϕ, and the bank angle μ, (as suggested in [12]) such that;

$$\sigma(\lambda,\phi,\mu) = \frac{\pi a^2 b^2 c^2}{(a^2 sin^2\lambda_e cos^2\mu_e + b^2 sin^2\lambda_e sin^2\mu_e + c^2 cos^2\lambda_e)^2} \quad (14)$$

$$\lambda_e = arccos\left(\frac{cos\phi}{cos\lambda}\right) \quad (15)$$

$$\mu_e = \mu - arctan\left(\frac{tan\phi}{sin\lambda}\right) \quad (16)$$

3. Theoretical Background On Hierarchical Reinforcement Learning

Reinforcement Learning (RL) can, informally, be defined as learning from incrementally gained experience via interactions with an environment to achieve a pre-defined goal. As such, it is a computational paradigm where an *agent* learns a policy of optimal *actions* by seeking to maximize the cumulative *rewards* obtained as a result of its interactions with an environment along a task horizon, in conjunction with a set of state that evolves during these interactions. The primary formal framework used for the specification of the above definition is the Markov Decision Process (MDP) [28, 7]. In a set theoretical setting, an MDP is formally defined (see, for instance, [31]) as a tuple of five elements $\langle S, A, T, R, \gamma \rangle$ where S is a finite set of states defining a discrete state space, A is a finite set of actions defining a discrete action space, T a transition function defined as $T: S \times A \times S \rightarrow [0,1]$, R is a reward function defined as $R: S \times A \times S \rightarrow R$ and $\gamma \in (0,1)$ denotes a discount factor. The transition function T and the reward function R together define the model of the MDP. Given an MDP, a policy is a computable function that outputs for each state $s \in S$ an action $a \in A$ (or $a \in A(s)$). Formally, a deterministic policy π is a function defined as $\pi: S \rightarrow A$. It is also possible to define a stochastic policy as $\pi: S \times A \rightarrow [0,1]$ such that for each state $s \in S$, it holds that $\pi(s,a) \geq 0$ and $\sum a \in \pi(s,a) = 1$.

It has been argued that RL as a computational paradigm faces some important challenges such as sampling and exploration inefficiency, sparse reward problem in long horizon tasks, low interpretability, inherent difficulties with transfer of learned skills (of the agent), complexity of reward function design process etc. (see for instance[41],[42],[43] for more discussion). HRL extends the capabilities of RL by introducing an approach where complex tasks are abstracted into smaller sub-tasks which can also be re-used for other similar complex tasks. More specifically, HRL increases exploration efficiency via temporally extended exploration (due to temporal correlation of higher level tasks), alleviates the sparse reward problem by utilizing various forms of intrinsic motivation in order to provide additional denser reward signals for individual abstractions, addresses sample inefficiency through efficient use of temporal and state abstractions, facilitate transfer learning and increase interpretability using various forms of abstractions.

HRL is formalized on the basis of the theory of Semi-Markov Decision Process (SMDP) [39], [40]. It is stated in [[39], [40] that an SMDP is a stochastic control process comparable to an MDP, but it differs from MDP in that it incorporates the concept of time for which an action is executed after it has been chosen. In the context of HRL, the actions which are not atomic (i.e., span a certain amount of time) are the subtasks. Starting from a state $s_t \in S$, assume that an agent chooses a subtask $\omega_t \in \Omega$, where Ω is the set of subtasks (or the subtask space). The reward obtained as a result of performing the subtask ω_t starting from state s_t is denoted as $R(s_t, \omega_t)$, calculated as follows,

$$R(s_t,\omega_t) = \mathbb{E}_{a\sim\pi_{\omega_t}(s)}\left[\sum_{i=0}^{c_{\omega_t}-1} \gamma^i r(s_{t+i}, a_{t+i}) \mid s_t, a_t = \pi_{\omega_t}(s_t)\right]. \quad (17)$$

Equation (17) indicates that the reward $R(s_t, \omega_t)$ is equal to the expected cumulative reward obtained while following the subtask policy π_{ω_t} from time t until the termination of ω_t after c_{ω_t} timesteps. Now, an optimal task policy would be the one that leads to the following desired maximum Q-value:

$$Q(s_t,\omega_t) = R(s_t,\omega_t) + \sum_{s_{t+c_{\omega_t}}, c_{\omega_t}} \gamma^{c_{\omega_t}} P\left(s_{t+c_{\omega t}}, c_{\omega_t} \mid s_t, \omega_t\right) \max_{\omega_{t+\omega_t}} Q\left(s_{t+c_{\omega_t}}, \omega_{t+c_{\omega_t}}\right), \quad (18)$$

Where, $\forall s \in S$ and $\forall \omega \in \Omega$. It should be noted that the Q-value in Equation (18) also depends on $R(s_t, \omega_t)$ and $P\left(s_{t+c_{\omega_t}}, c_{\omega_t} \mid s_t, \omega_t\right)$. These two quantities are determined by the execution of ω_t using its policy π_{ω_t}. Therefore, an agent actually needs to learn multiple policies at different levels of a task decomposition hierarchy.

As noted in [41], earlier HRL methods, were mostly based on three main approaches: the options formalism of Sutton et.al,[47] the Hierarchies of Abstract Machines (HAMs) of Parr [39] and Parr and Russell [40], and the MAXQ framework of Dietterich [48], which share many common elements and more importantly rely on SMDP as the underlying theory. Further research has resulted in many variants and improvements as detailed in recent comprehensive surveys such as [42] and [43]. One such relatively recent approach listed in the larger taxonomy of approaches is called Feudal Reinforcement Learning (Feudal RL) [49]. Feudal RL involves a hierarchy, in which the action space of a higher-level policy consists of subgoals corresponding to various subtasks. A subgoal chosen by the higher-level policy is taken as input by a universal policy at the lower level. The objective of this lower-level universal policy is to achieve the input subgoal. The universal policy at each level can be treated as a sub-agent (as part of the HRL agent) that can perform all the possible subtasks at that level. This leads to the feudal concept of a "manager" sub-agent (higher-level policy) directing a "worker" sub-agent (lower-level policy). This method is particularly relevant to our work, since we adopt the main framework proposed in it.

3.1 Methodological Details of Our Framework Based on HRL

In this work we adopt a two-layered architecture consisting of a controller and a meta-controller. We adopt a Feudal RL approach which is briefly described as follows: a higher-level manager selects a subtask that is to be fulfilled by a worker at a lower-level. The manager assigns the subtask to the worker via a sub-goal. The objective of the worker is to achieve the designated sub-goal. The cumulative rewards produced by a task are visible only to the manager at the highest level while the workers at other levels learn using the rewards for reaching the sub-goals. In this setting, we use a multinomial logistic regression model [51] to implement a meta controller for the manager which is then trained using a state transition graph that models the selection of the appropriate high-level goal based on a meta state depicting the status of the higher-level learning process. Since this meta state is a categorical description of the higher-level tasks and the confidence in the learning level of those tasks, we use an encoding scheme similar to one-hot encoding to model the relevant state data.

For the worker, we implement a lower-level controller, using one of the most popular model-free, Q-Learning-based RL algorithms, namely Deep Q Network (DQN) which was developed by [20]. DQN relies on a neural network for approximating the behavior of the Q-function in a non-linear manner. In fact, a DQN is a multi-layered (deep) neural network which takes a vector of observations, $\omega_t \in \Omega$, as its input, and assigns likelihood values to a vector of action values $Q(\omega_t, .; \theta)$ in its output, where θ are the parameters of the network. Standard Q-learning algorithms were plagued with instability until [20] introduced two mechanisms that address this deficiency. The first of these mechanisms is the *experience replay* which involves accumulating a customizable window of observation sequence and randomizing over the data in this sequence to remove correlations in the observation history and to smooth over changes in the data distribution. During the training of the DQN, experiences are first accumulated in the experience replay buffer using ϵ-greedy policies. The second of these mechanisms is called a *target network* and relies on the use a separate neural network (architecturally identical to the *value network*), with parameters θ^- . During the training, the parameters of the value network are copied over to the target network every k steps, so that then $\theta_t^- = \theta_t$, and kept fixed on all other steps.

In its original form, experience replay buffer is sampled uniformly. A more effective variant was introduced by [25] called *prioritized experience replay*. The motivation here is to increase the impact of surprising and/or task-relevant transitions within the experiences during the learning process. More specifically, this technique, uses the magnitude of the temporal-difference (TD) error pertaining to transitions, and tends to favor (i.e., prioritize) those with high expected learning progress. Such selective prioritization may result in a loss of diversity and can introduce bias, but these are reconciled with stochastic prioritization and importance sampling respectively.

4. Casting the Problem into A Reinforcement Learning Context

RL assumes a specific framework in which an agent interacts with its environment conforming to a relatively simple but a well-defined pattern. Hierarchical RL, on the other hand, adds a further meta-layer for the assignment and management of tasks that may consist of an arbitrary hierarchy. The structural (or data-model centric) aspects of this interaction scheme and its associated management layer can be specified via an agent observation space, a meta-layer observation space and an action and task space. The algorithmic (or behavioral) aspects, on the other

hand, can be specified via the reward function. In this section we first provide the details of those structural and behavioral aspects, and then we proceed by introducing our algorithm that implements the actual hierarchical learning scheme based on the specified aspects.

4.1 Agent Observation Space

An observation for the lower-level controller agent, $\rho_t \in \Omega$, is the observable part of the environment state by the agent at time step t. ω_t is defined as a tuple as below:

$$R_g = \sqrt{(x_g - x)^2 + (y_g - y)^2}, \tag{19}$$

$$\alpha = arctan\left(\frac{y_g - y}{x_g - x}\right), \tag{20}$$

$$\beta = |\psi - \alpha|, \tag{21}$$

$$\rho_t = <\| x \|, \| y \|, \| R_g \|, P_{tr}, P_k, \alpha, \psi, \mu, \beta, \| RCS_1 \|, \ldots, \| RCS_n \|, \ \mathrm{MS_g} > \tag{22}$$

where P_{tr} is the aggregate probability of track at a certain time step, as defined earlier in Equation (4), $P_k(t)$ is the probability of kill as defined earlier in Equation (5), x_g, y_g are the coordinates of the center of the goal area in x-y plane, x, y are the coordinates of the UAV in x-y plane, α is the angle between the x-axis and the line connecting the current UAV position to the center of the goal area (i.e. the goal angle), β is the difference between the UAV direction angle (i.e. ψ as defined in Equation (8)) and the goal angle, μ is the UAV bank (or roll) angle as defined in Equation (12), and RCS_i is the RCS of the UAV with respect to the i^{th} radar in the mission area, n being the total number of radars deployed in the mission area. $\mathrm{MS_g}$ is the one hot encoded sub-goal achievement meta state that describes the current state of the agent with regard to achieving the set of all sub-goals. The $\|.\|$ symbol denotes normalization.

4.2 Meta-Level Observation Space

The observation space for higher level meta controller, σ_t, is an n-tuple representing the task (or goal) meta state, encoded similar to one-hot encoding defined as follows;

$$\sigma_t = < \mathrm{x_{1,}} \ldots \mathrm{x_n}, \mathrm{m} >; \ \mathrm{n} = number\ of\ subgoals$$

$$x_i = 1_M(u_i) = \begin{cases} 1 & when\ u_i \in G_t \\ 0 & when\ u_i \notin G_t \end{cases}$$

$$m = \begin{cases} 1 & when\ x_i\ is\ learned\ successfully \\ 0 & when\ x_i\ is\ \ \ NOT\ learned \end{cases}$$

$$G_t = \{g_{1_t}, \ldots, g_{k_t}\}; \mathrm{k} \leq \mathrm{n}; \ \mathrm{k_t} = number\ of\ subgoals\ achieved\ by\ the\ agent\ at\ time\ t$$

where $1_M(u_i)$ is an indicator function that determines whether a specific element of the tuple σ_t is 1 or 0 depending on the sub-goal number being encoded; m is a binary code that indicates whether the encoded goal is learned successfully or not, and G_t is the set of sub-goals achieved (or visited) by the agent at time t including the final goal. G_t grows as the agent achieves more goals. Note that with this encoding scheme our framework supports categorical states, so it is inherently suitable for extension to other contexts, involving more liberal task definitions such as non-navigational behaviors (e.g., hover along a circle for reconnaissance).

4.3 Action And Task Space

For the lower-level controller, the actions in our problem setting are defined in terms of steering input, u, to the UAV navigation system, as specified in Equation (8). We adopt a discrete action space which is defined as follows:

$$A = \{u_1, \ldots, u_7\}, \quad u_i \in [-15,15] \tag{23}$$

$$u_i = u_{i-1} + 5, \tag{24}$$

where A is a finite set that defines all the available actions containing 7 elements. The elements range from -15 to 15 with an increment of 5. Positive values cause the aircraft to perform a right-turn maneuver, whereas negative values cause the aircraft to perform a left-turn maneuver.

For the higher-level meta controller, the task space consists of a single integer representing the task (i.e., sub-goal) number to be assigned to the lower-level worker as the target.

4.4 Reward Function

Our reward function is designed to provide direction and range feedback with respect to the designated (active) goal, as early as possible The overall cumulative reward signal, R_t, is composed of a number of sub-components as defined below:

$$RW_{gr} = \| 1 - R_g \|$$

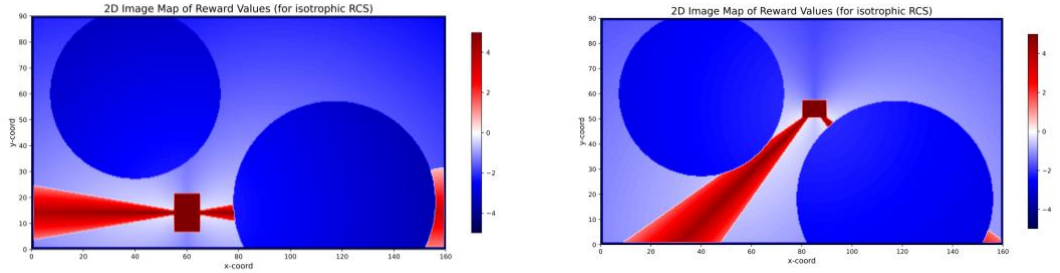
$$PN_{ptk} = \begin{cases} 0, & \textbf{if } P_{tr} < \xi \\ 2\big(P_{tr}(t) + P_k(t)\big), & otherwise \end{cases}$$

$$FB_{dir} = \begin{cases} 5(1 - \| \beta \|), & \textbf{if } \beta < 10 \\ -2 \| \beta \|, & otherwise \end{cases}$$

$$R_t = \begin{cases} -50, & \textbf{if } (P_k(t) > \zeta) \vee (UAV\ out\ of\ area) \\ \alpha \mathrm{RW}_{\mathrm{sg}}, & \textbf{if } UAV\ in\ goal\ area \\ RW_{gr} + FB_{dir} - PN_{ptk}, & otherwise \end{cases}$$

where P_{tr} is the aggregate probability of track at a certain time step, as defined in Equation (4), $P_k(t)$ is the probability of kill which was defined in Equation (5), ξ is the tracking probability threshold (the probability value at which a track lock is initiated), ζ is the kill probability threshold (i.e. the value of $P_k(t)$ for which a UAV-kill event is set to happen), R_g and β are defined in Equations (19) and (21) respectively, PN_{ptk} is the penalty for being tracked or killed, RW_{gr} is the reward for getting nearer to the goal, FB_{dir} is the direction feedback which increases as the heading of the UAV aligns better towards the goal, $\mathrm{RW}_{\mathrm{sg}}$ is a specific numeric reward for reaching a certain sub-goal weighted with a coefficient α. Here α takes greater values as the target sub-goal is closer to the final goal.

A 2D visualization scheme, as depicted in Figure 2, is employed to represent the output of the reward function, where reward values are used to generate a color-coded image map. The figure illustrates a rectangular mission area containing 2 radars installed in various locations, and some interim goal areas placed in different locations within the mission area. Radar coverage areas are indicated via opaque circular regions. As it would be expected, those circular areas are discouraged by the reward function, so blue color signifies negative reward values (i.e., punishment), darker shades indicating more dramatic penalty levels. Positive reward values are indicated by a red color, signifying direction-wise and range-wise proximity of the agent to active (i.e., currently assigned) goals. In this respect, Figure 2 (a) depicts a case where an agent with a direction angle of 0 (i.e., $\psi = 0$) obtains increasingly higher positive rewarded values as it moves towards the goal along a straight line. Note that the central line of the



(a) ψ = 0, target is Goal-1 (b) ψ = 45, target is Goal-2

Figure 2 Color-coded representation of reward function returns for different UAV direction angle (ψ) values. (ψ equals 0 and the target is Goal-1 on the left. ψ equals +45 and the target is Goal-2 on the right)

path exhibits highest positive rewards (indicated by a darker red color), since it minimizes the difference between the goal angle and UAV's direction. Figure 2 (b) illustrates a case where an agent with a direction angle of 45 (i.e., $\psi = 45$) gets increasingly higher positive reward values as it approaches towards the goal along a diagonal line with a 45 degrees angle with the center line of the goal area etc. Note that the illustrations are only valid for scenarios employing an isotropic RCS for the UAV. Cases involving non-isotropic RCS, would result in dramatic changes (in fact fluctuations) in the probability of detection values, depending on the maneuvers of the UAV, due to changes in the three engagement angles. This would make the reward signals much harder to map in a comprehensive way.

4.5 The Hierarchical RL Algorithm

We now describe the algorithm that implements the HRL solution. The procedure given in Algorithm 1 starts with the initialization of the meta controller using the set of goals and a multinomial logistic regression model. Then there is a pre-training step (in line 6) in which the meta controller is trained using a set of meta state transitions and corresponding next states for each meta state tuple in this set. Note that the state tuples are n-tuples encoded using a scheme similar to one-hot encoding and the next states are task (or goal) numbers denoting the next task (goal) to be targeted. For instance, the following examples illustrate three cases where in the first one, the first goal was visited but wasn't learned (so the last member of the tuple, i.e., the confidence bit m, is still 0) therefore the next meta state (goal) is 1, suggesting that the agent should still target this goal until it confidently learns it. In the second one, the meta state indicates that the first goal is visited and learned, so the next task is targeting goal number 2. In the third example, the first goal and the second goal are visited and the last goal (i.e. goal number 2) is also learned, so the next goal is goal number 3, etc. Note that these examples are valid for a 3-goal scenario. Note also that, contrary to strict one-hot encoding, meta state tuple can contain multiple "1" values.

$(1,0,0,0) \rightarrow 1$
$(1,0,0,1) \rightarrow 2$
$(1,1,0,1) \rightarrow 3$

Following the training of the meta controller the environment is reset and low-level training of the controller starts as a conventional RL training loop, except that the outer loop uses the *pre-trained meta controller* to assign high level goals each time a specific goal is achieved and learned successfully. This is indicated in line 12 of the algorithm. The loop starting with line number 14 handles the training of lower level controller, which involves asking the controller to predict the next low level action (line 15), forwarding the environment state one step using the new action (line 16), calling the optimization step to perform training of the Q-Function which is implemented as a DQN in our case (line 18), checking if a goal was reached and making sure the active goal (i.e. the goal that was assigned to the agent currently) was learned successfully (lines 19 to 29). Note that this procedure is a two-stage training process where the first stage is an off-line pretraining phase (for the meta controller). Note also that this framework does not allow for dynamic goal discovery, the meta level goal structure has to be predefined.

Algorithm 1 Procedure for training the hierarchical DQN agent

```
1  Procedure HRL (X,Y,G,F_lr,env)
2  //X is a set of meta state tuples, Y is the next state vector (for a given
3  meta state)
4     initialize meta_controller(G,F_lr)//F_lr is a sklearn logistic regression model
5                                       // G is a set of goals
6     meta_controller.train(X, Y) //train the logit model using X and Y
7     obs = env.reset()
8     R = 0; t = 0
9     While t < max_steps
10       goal_state = obs.get_metastate()
11        //get next task (goal) from the meta controller
12       new_goal = meta_controller.get_next_goal(goal_state)
13       env.set_target_goal(new_goal)
14       While True
15          act = contoller.act(obs)  //get next action from the controller
```

```
16          obs, rew, done = env.step(act)
17          R += rew; t += 1
18          contoller.train(obs, rew, done)
19          if goal_reached
20             if is_active_goal_learned()
21                env.designate_next_goal()
22             else:
23                success_ratio = evaluate_success(new_goal)
24                if success_ratio >= 0.7
25                   learned_current_goal = True
26                   env.set_confidence(new_goal,1)
27                else:
28                   learned_current_goal = False
29                   obs = env.reset()
30          if done or reset or learned_current_goal
31             break
32       End While
33    End While
34 End Procedure
```

5. Experiments and Results

The experiments are carried out in a simulated tactical area where two radars are installed. The x and y dimensions of the area is 160 km by 90 km. The radars have different performance specifications characterized by the parameters given in Table *1*, where P_{tx} refers to peak transmit power, f is the operating frequency, F_n irefers to the noise figure, pw is the pulse width, P_{fa} is the probability of false alarm, SNR_{min} is the minimum required SNR, t_{resp} is the response time of the radar and v_m is the speed of the Surface to Air (SAM) missile. Both radars have the ability to perform non-coherent pulse integration, k and l parameters given in Equation (2) are taken to be $k = 30, l = 4$. Three scenarios are experimented:

1. In the first one only one goal exists in the environment which is located far from the entry point to the mission area. In this environment a standard DQN algorithm is used to illustrate the results of using plain RL in such long horizon tasks.
2. In the second scenario a two-goal environment is used and the agent is trained using HRL approach.
3. In the third scenario one more sub-goal is added to the environment and the performance of the HRL agent is demonstrated.

Table 1 Important parameters of two different radar types

Radar Type	$P_{tx}(dBW)$	$f(GHz)$	$F_n(dB)$	$pw(\mu s)$	P_{fa}	SNR_{min} (dB)	$t_{resp}(s)$	$v_m(m/s)$
Type-1	54.47	8	0.4	8	10^{-6}	10	22	1200
Type-2	55.18	7	0.4	5.19	10^{-6}	13	20	1300

In all of the scenarios above;

1. The UAV is assumed to cruise at a constant altitude of $9000m$, at a constant speed of $170m/s$ $(0.49Mach)$. In other words, take-off, climbing, descending and landing phases are not considered.
2. The target fluctuation category of the UAV is assumed to be swerling-1 as defined in [29].
3. At each time step in the simulation, the aggregate probability of track (obtained using Equation (4)) is calculated and checked against a *tracking probability threshold* of value 0.9. If the aggregate prob. of track is greater than that value, then for each of the contributing radars, a track sequence is triggered at that time step.
4. If a track sequence is initiated for a particular radar, then missile hit time, $t_{hit} = R_t/v_m$ is calculated for that radar and at each consecutive time step the probability of track is checked against a *track loss threshold*. If the probability of track continues to remain above the track loss threshold for a time window equal to $t_{resp} +$

t_{hit} of that radar, then the UAV is considered to be lost. Note that it is assumed that t_{hit} stays constant during the threat window.

5. To model the difficulty of breaking a track lock via maneuvers once it is acquired, the track loss threshold is set to 0.6, indicating that the track lock remains unbroken until the probability of track goes below 0.6.

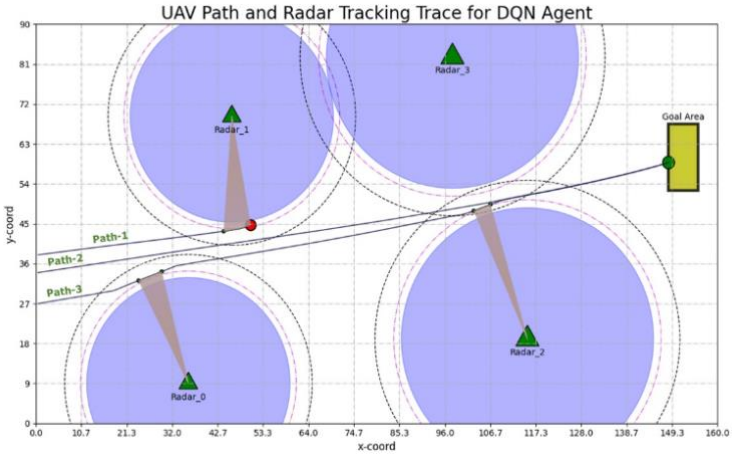


Figure 3 An example post mission trace graph

During the experiments, the flight path of the aircraft and the track history of the radar are logged and plotted to provide a visual performance evaluation scheme. An example graph is given in Figure 3. Here, opaque circular areas delineate the region where probability of track is greater than 0.9 for an isotropic RCS of $1.2m^2$. The dotted circles at the outer perimeter indicate the points at which the probability of track is equal to 0.54. Maximum detection range based on the minimum SNR value of each radar is also calculated and illustrated using red dotted circles in between of those two. Some example flight trajectories are included in the figure, where Path-1 denotes a scenario where the UAV was destroyed by Radar-1 following a track period (track initiation point and kill point are denoted by a small black dot and relatively larger red dot respectively), Path-2 illustrates a fully successful trial and Path-3 illustrates two tracking events initiated by Radar_0 and Radar_2, neither of them being able to destroy the target. This layout and notational convention are used throughout the experiments.

For all scenarios, to model the lower-level controller we use a DQN with input size of 15 (equal to the size of the observation vector), an output size of 7 (equal to the size of the discrete action space) and 2 hidden layers of 400 channels each. We use ADAM [14] algorithm for the optimization of the neural network. The experience replay buffer has a start size of 256, and is sampled using a batch size of 128. The exploration policy is managed via a decaying ϵ-greedy explorer epsilon values ranging in the interval [1.0, 0.1].

5.1 Presentation of The Results and Discussions

In all of the experiments we performed the evaluation process at 10 episode intervals, averaging cumulative reward obtained from 5 evaluation runs. During evaluation runs the agent does not take any exploratory actions, it fully exploits the current learning level of the DQN. As illustrated in , in the first scenario the agent was unable to learn to reach the target goal area although the experiment was extended to last for nearly 700 episodes (around 1 million steps). Note that the training curve seem to stabilize around a particular cumulative reward value (around 7000), but evaluation curve swings along a large interval. This is because the DQN agent struggles to converge to a successful policy given the delayed feedback conditions in this long horizon setting.

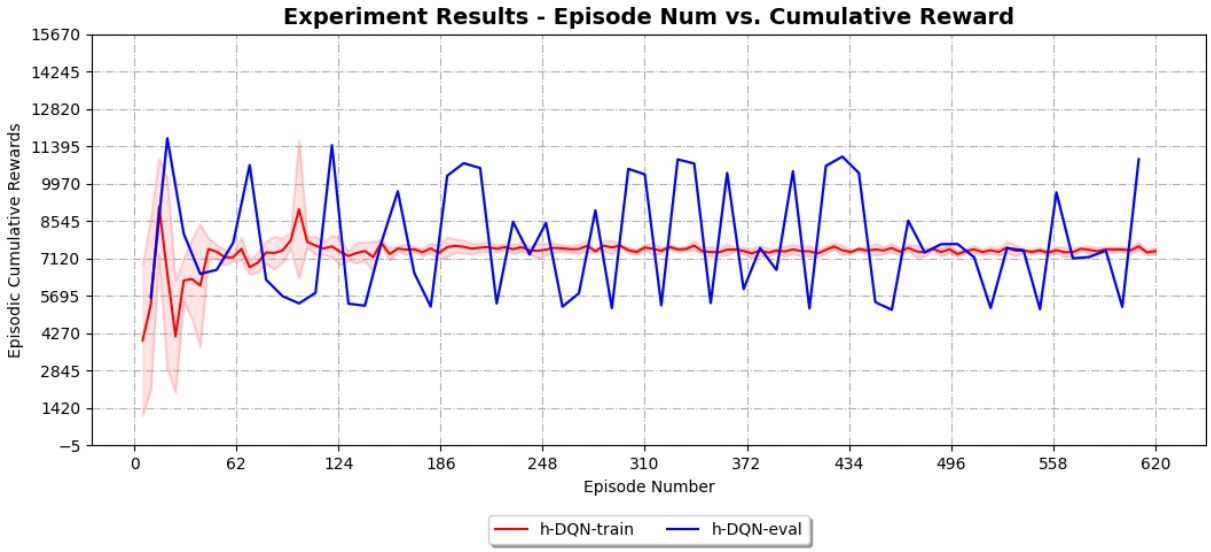
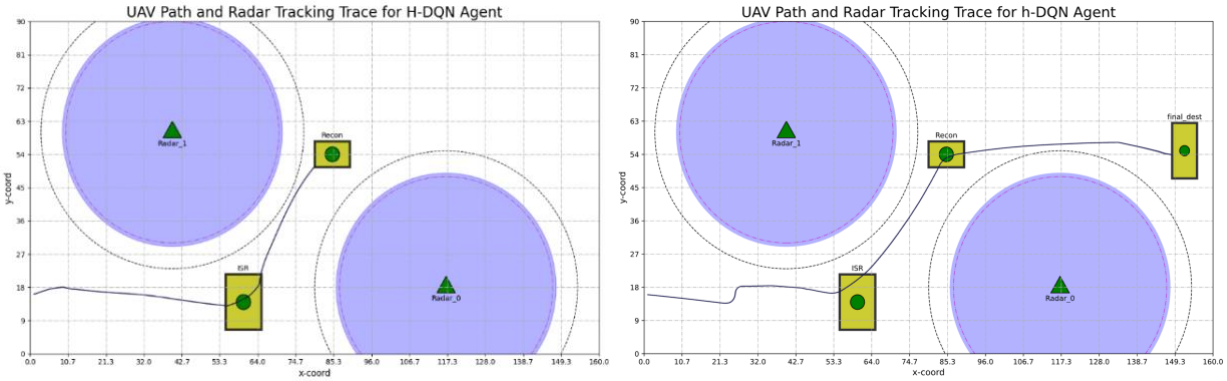


Figure 4 Training statistics and evaluation results for the first scenario



(a) Two goal setting (b) Three goal setting

Figure 5 İllustration of navigation paths for the second and the third scenario

Figure 5 on the other hand illustrates the path of the UAV learned via the HRL algorithm, in two-goal (a) and three-goal (b) settings. The labels attached to the goal areas stand for Intelligence Surveillance and Reconnaissance (ISR) and Reconnaissance (RECON) to indicate that interim goals may serve different purposes in the mission planning.

Figure 6 provides the training statistics and evaluation results for the second scenario (i.e. two-goal setting). Note that small green circles that appear on the evaluation curve denote the point where the agent successfully learns the actively assigned goal. So, for instance, for the two-goal setting the first goal is learned at the end of 10th episode, whereas the second (and final) goal was achieved at the end of episode number 100.

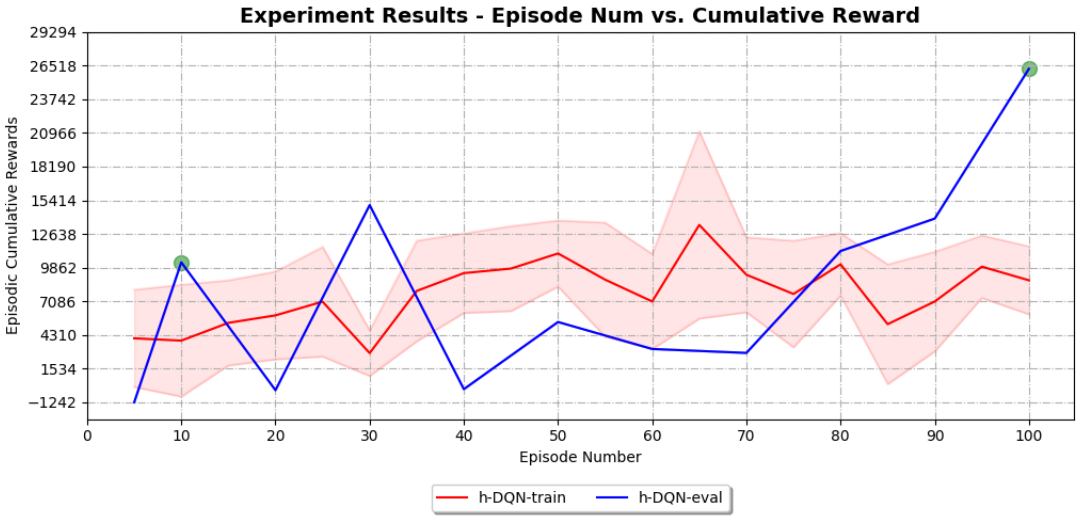


Figure 6 Training statistics and evaluation results for two-goal scenario

Similarly, Figure 7 provides the training statistics and evaluation results for the third scenario (i.e. three-goal setting). Note that this time the first goal is achieved in episode number 13, the second goal is achieved in episode number 250, and the third goal is achieved around episode number 290. Note that in this three-goal scenario the second goal is learned much later compared to the two-goal scenario. Although we use fixed seed to initialize random number generators used in the framework, as we pointed out earlier, we use perturbation of the UAV entry point to the mission area, which introduces a certain degree of randomness to the process. So this behavior is something we would expect.

The results indicate that our HRL framework ensures that the agent learns the complex navigation arrangements in a reasonable amount of time, and via a well-defined procedure.

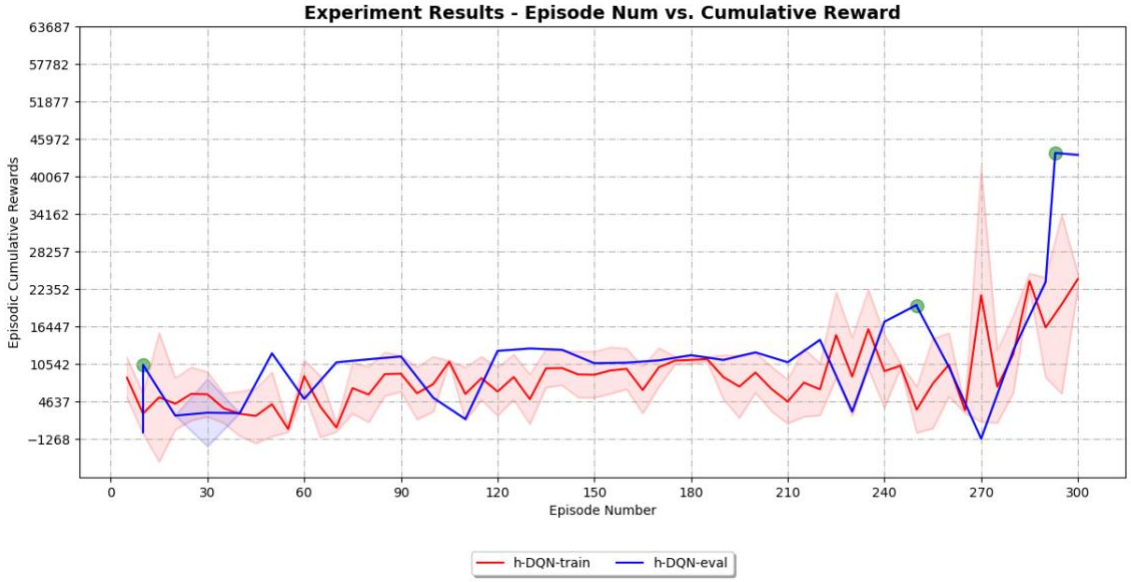


Figure 7 Training statistics and evaluation results for three-goal scenario

6. Conclusion

In this paper we have implemented an HRL based framework to provide a solution to sparse reward problem surfacing in complex, long-horizon path planning settings. Our proposed framework combines a meta controller for higher level goal assignment and a controller that determines the lower-level actions of the agent in a novel way, and thus constitutes one of the first examples of such a hierarchical learning scheme put into the use for UAV path planning in tactical environments. Several concluding remarks are worth noting:

1. The experiment results are a clear manifestation of the fact that hierarchical learning schemes do indeed lead to considerable improvement of the performance of the RL agent in path planning scenarios.
2. This two-layer framework ensures that an optimal plan for a complex path, organized as multiple goals, is achieved gradually, through piecewise assignment of sub-goals, and thus as a result of a staged, efficient and rigorous procedure.
3. The fact that our meta layer is pre-trained using a set of pre-defined state transitions that prescribe the evolution of goal assignment based on well-defined goal achievement state, makes the higher-level logic deterministic. This brings about performance and more robustness to the training process but limits the applicability of the solution to relatively more conservative settings. This implies that the more liberal problem settings such as those that require dynamic goal discovery are not readily addressed.
4. Our framework supports categorical states at the meta-level, so the proposed encoding scheme combined with a variable reward function selection mechanism is inherently suitable for extension to other similar contexts, involving more liberal task definitions such as non-navigational behaviors (e.g., hover along a circle for reconnaissance).

Acknowledgements

The work presented in this paper has been partially supported by TÜBİTAK BİLGEM. We are grateful for that support. All of the theoretical and practical work including authorship of the paper is performed by the single author.

References

[1] Abell DC, Caraway III WD. A method for the determination of target aspect angle with respect to a radar, July, 1998.
[2] Bertsekas DP. Reinforcement Learning and Optimal Control. Athena Scientific, Belmont, Massachusetts.
[3] Bouhamed O, Ghazzai H, Besbes H, Massoud Y. Autonomous uav navigation: A ddpg-based deep reinforcement learning approach, 2020.
[4] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W. Openai gym. arXiv preprint arXiv:1606.01540, 2016.
[5] Challita U, Saad W, Bettstetter C. Deep reinforcement learning for interference-aware path planning of cellular-connected uavs. In 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–7.
[6] Fujita Y, Nagarajan P, Kataoka T, Ishikawa T. Chainerrl: A deep reinforcement learning library. Journal of Machine Learning Research 2021; 22(77): 1–14.
[7] Garcia F, Rachelson E. Markov Decision Processes, pp. 1–38. John Wiley and Sons, Ltd, 2013.
[8] Gosavi A. Control Optimization with Reinforcement Learning pp. 197–268. Springer US, Boston, MA, 2015.
[9] Hare J. Dealing with sparse rewards in reinforcement learning. CoRR, abs/1910.09281, 2019.
[10] Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, Sendonaris A, Dulac-Arnold G, Osband I, Agapiou JP, Leibo JZ, Gruslys A. Learning from demonstrations for real world reinforcement learning. CoRR, abs/1704.03732, 2017.
[11] Inanc T, Muezzinoglu MK, Misovec K, Murray RM. Framework for low-observable trajectory generation in presence of multiple radars. Journal of Guidance Control and Dynamics 2008; 31(6):1740–1749.
[12] Pierre T, Kabamba, Semyon M, Meerkov, Frederick H. Zeitz. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking. Journal of Guidance Control Dynamics 2006; 29(2):279–288.
[13] Kang EW. Radar System Analysis, Design and Simulation. ARTECH HOUSE, INC. 2008.
[14] Kingma DP, Ba J. Adam: A method for stochastic optimization, 2017.
[15] Aristotelis L, Anestis F, Ioannis V. Deep reinforcement learning: A state-of-the-art walkthrough. The Journal of Artificial Intelligence Research 2020; 69: 1421–1471.

[16] Le TP, Vien NA, Chung T. A deep hierarchical reinforcement learning algorithm in partially observable markov decision processes. IEEE Access 2018; 6:49089–49102.
[17] Jeong-Won L, Bruce W, Kelly C. Path Planning of Unmanned Aerial Vehicles in a Dynamic Environment.
[18] Mahafza BR. Radar Systems Analysis and Design Using Matlab. CRC Press, third edition, 2013.
[19] Mes MRK, Rivera AP. Approximate Dynamic Programming by Practical Examples, Springer International Publishing, Cham. pp. 63–101.
[20] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015; 518(7540): 529–533.
[21] Pelosi M, Kopp C, Brown M. Range-limited uav trajectory using terrain masking under radar detection risk. Appl Artif Intell 2012; 26(8): 743–759.
[22] Pham HX, La HM, Feil-Seifer D, Nguyen LV. Autonomous UAV navigation using reinforcement learning. CoRR, abs/1801.05086, 2018.
[23] Qu C, Gai W, Zhong M, Zhang J. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (uavs) path planning. Applied Soft Computing 2020; 89: 106099.
[24] Benjamin R. A tour of reinforcement learning: The view from continuous control, Annu Rev Control Robot Auton Syst 2019; 2(1): 253–279.
[25] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. cite arxiv:1511.05952Comment: Published at ICLR 2016.
[26] Skolink MI. Radar Handbook. McGraw-Hill, second edition, 1990.
[27] Sutton RS, Barto AG, Williams R J. Reinforcement learning is direct adaptive optimal control. IEEE Control Systems Magazine 1992;12(2):19–22.
[28] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA, 2018.
[29] Swerling P. Probability of Detection for Fluctuating Targets. RAND Corporation, Santa Monica, CA, 1954.
[30] Mirco T, Harald B, Richard N, David G, Marco C. Uav path planning using global and local map information with deep reinforcement learning, 2020.
[31] Martijn van Otterlo and Marco Wiering. Reinforcement Learning and Markov Decision Processes, chapter 1, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
[32] Chao Y, Xiaojia X, Chang W. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. J Intell Robot Syst 2020; 98(2): 297–309.
[33] Michael Z, Stan U, Robert M. Aircraft routing under the risk of detection. Naval Research Logistics 2006; 53(8):728–747.
[34] Frederick H. Zeitz. Ucav path planning in the pesence of radar-guided surface-to-air missile threats, Phd thesis, University of Michigan, 2005.
[35] Weiwei Z, Wei W, Nengcheng C, Chao W. Efficient uav path planning with multiconstraints in a 3d large battle field environment. Math Probl Eng 2014:597092.
[36] Zhe Z, Jian W, Jiyang D, Cheng H. Rapid penetration path planning method for stealth uav in complex environment with bb threats Int J Aerosp Eng 2020:8896357.
[37] Alpdemir MN. Tactical UAV path optimization under radar threat using deep reinforcement learning. Neural Comput Applic 2022; 34: 5649–5664.
[38] AlMahamid F, Grolinger K. Autonomous Unmanned Aerial Vehicle navigation using Reinforcement Learning: A systematic review Eng Appl Artif Intell 2022; 115: 105321
[39] Parr R. Hierarchical control and learning for Markov decision processes, Ph.D. Thesis, University of California at Berkeley, 1998.
[40] Parr R, Russell. Reinforcement learning with hierarchies of machines, in: Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, MA, 1998, pp. 1043–1049.
[41] Barto AG, Mahadevan S. Recent Advances in Hierarchical Reinforcement Learning Discrete Event Dyn Syst 2003; 13: 341–379.
[42] Hutsebaut-Buysse M, Mets K, Latré S. Hierarchical Reinforcement Learning: A Survey and Open Research Challenges, Mach Learn Knowl Extr 2022; 4(1): 172-221.
[43] Pateria S, Subagdja B, Tan A, Quek C. Hierarchical Reinforcement Learning: A Comprehensive Survey. ACM Comput Surv 2022; 54(5):35.
[44] Cheng Y, Li D, Wong WE, Zhao M, Mo D. Multi-UAV Collaborative Path Planning using Hierarchical Reinforcement Learning and Simulated Annealing J Int J Performability Eng 2022;18(7): 463-474.
[45] Qin Z, Zhang X, Zhang X, Lu B, Liu Z, Guo L. The UAV Trajectory Optimization for Data Collection from Time-Constrained IoT Devices: A Hierarchical Deep Q-Network Approach. Applied Sciences. 2022; 12(5): 2546.
[46] Li B, Wu Y, Li G. Hierarchical reinforcement learning guidance with threat avoidance, Journal of Systems Engineering and Electronics 2022; 33(5): 1173-1185.
[47] Sutton RS, Precup D, Singh S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artif Intell 1992; 112: 181–211.

[48] Dietterich TG. Hierarchical reinforcement learning with the MaxQ value function decomposition. J Artif Intell Res 2000; 13: 227–303.
[49] Dayan P, Hinton GE. Feudal reinforcement learning. Advances in Neural Information Processing Systems. Morgan-Kaufmann 1993; 5: 271–278.
[50] Qin Z, Zhang X, Zhang X, Lu B, Liu Z, Guo L. The UAV Trajectory Optimization for Data Collection from Time-Constrained IoT Devices: A Hierarchical Deep Q-Network Approach. Applied Sciences 2022; 12(5):2546.
[51] Hosmer DW, Lemeshow S. Applied Logistic Regression, John Wiley & Sons, Inc., Second Edition, 2000.