Resource Allocation for Multi-target Radar Tracking via Constrained Deep Reinforcement Learning

Ziyang Lu and M. Cenk Gursoy

***Abstract*—In this paper, multi-target tracking in a radar system is considered, and adaptive radar resource management is addressed. In particular, time management in tracking multiple maneuvering targets subject to budget constraints is studied with the goal to minimize the total tracking cost of all targets (or equivalently to maximize the tracking accuracies). The constrained optimization of the dwell time allocation to each target is addressed via deep Q-network (DQN) based reinforcement learning. In the proposed constrained deep reinforcement learning (CDRL) algorithm, both the parameters of the DQN and the dual variable are learned simultaneously. The proposed CDRL framework consists of two components, namely online CDRL and offline CDRL. Training a DQN in the deep reinforcement learning algorithm usually requires a large amount of data, which may not be available in a target tracking task due to the scarcity of measurements. We address this challenge by proposing an offline CDRL framework, in which the algorithm evolves in a virtual environment generated based on the current observations and prior knowledge of the environment. Simulation results show that both offline CDRL and online CDRL are critical for effective target tracking and resource utilization. Offline CDRL provides more training data to stabilize the learning process and the online component can sense the change in the environment and make the corresponding adaptation. Furthermore, a hybrid CDRL algorithm that combines offline CDRL and online CDRL is proposed to reduce the computational burden by performing offline CDRL only periodically to stabilize the training process of the online CDRL.**

***Index Terms*—Constrained optimization, extended Kalman filter, multi-target tracking, radar, reinforcement learning, resource allocation.**

I. Introduction

Radar is an active remote-sensing technique that is widely used for detection, tracking, and surveillance in various applications including e.g., remote sensing of the environment, space exploration, aircraft navigation, air/sea traffic control, law enforcement, military operations, etc. With the emergence of autonomous vehicles and drones, radar operation in challenging environments has grown further in importance. How to efficiently allocate radar resource becomes critical and has been addressed extensively in the literature. The resource allocation problem has been addressed with conventional optimization approaches in [1] and [2]. Another line of work in the literature is based on game theoretical approaches. For instance, the authors in [3] formulate the power allocation problem in a multi-radar system as a non-cooperative game and perform an analysis of the Nash equilibrium and its convergence. In [4], an overview of cognitive radar concepts is provided, and three hierarchical levels of the cognitive radar architecture, namely extracting the information, resource management, and refining the knowledge of the environment, are highlighted. The authors subsequently present a survey of the development of radar resource management techniques, each with increasing adaptivity, leading up to the recognition of the term cognitive radar. For more details on studies on active tracking, adaptive tracking, and cognitive radar resource management techniques, we refer to [4] and references therein.

Radar environment is typically nonstationary and this has a significant impact on the radar returns (i.e. echoes). Due to this, it is desired to design a cognitive radar that can keep updating its knowledge of the surrounding environment and make dynamic decisions to adapt to the environment. Cognitive radar is a rapidly developing field that utilizes artificial intelligence (AI) techniques to improve the performance of radar in complex and dynamic environments. Related work of cognitive radar includes various approaches such as machine learning, game theory, and cognitive radio [5]. An overview of cognitive radar systems is provided in [6], in which the focus is on signal processing, dynamic feedback from the receiver, and preservation of the information in a cognitive radar system. We note that cognitive radar techniques enable the development of multifunction radar, which, equipped with arbitrary waveform generation and electronic beam steering, is capable of supporting multiple radar functionalities [4], including multi-target tracking.

A. Related Work

Several recent studies considered resource allocation in radar systems. For instance, the authors in [7] address time allocation in multi-target tracking with the method of extended Kalman filter and the framework of partially observable Markov decision processes and policy rollout. In [8], model predictive control (MPC) is employed for time allocation in a radar system. Simulation results show that policy rollout and MPC enable the effective determination of both the revisit interval and the dwell time to reduce the variance of the estimations. Policy rollout and MPC are offline methods in which the decisions are made based on the experience in an environment generated following the prior statistical knowledge (e.g., regarding the measurement noise and maneuverability noise). However, radar environment is nonstationary and it

Ziyang Lu and M. Cenk Gursoy are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY (email: zlu112@syr.edu, mcgursoy@syr.edu)

may become difficult for offline algorithms to sense and determine sudden changes in the environment.

In such cases, an online approach that can adapt to the changes in the environment can be preferred. Recently, deep reinforcement learning (DRL) has been used to train agents in various complex decision-making scenarios and is shown to demonstrate strong performance [9]. DRL is a model-free algorithm with features making it suitable for solving problems in dynamic radar applications. Indeed, several recent studies have incorporated DRL in decision-making for radar applications. For instance, the work in [10] applies DRL to a spectrum allocation problem for multi-target detection, where it is shown that DRL is able to improve the performance of detection while mitigating interference to the other coexisting wireless system. Other related DRL work in radar environment has been conducted in [11] and [12].

In this paper, we formulate the multi-target tracking problem as a constrained Markov decision process (CMDP). Under a budget constraint, it is often desired to describe the problem as a CMDP and have one cost minimized while keeping the other types of costs constrained subject to certain specific requirements [13]. Recent learning-based characterizations and results related to decision-making within a CMDP can be found in [14], [15], [16], [17] and [18]. Specifically, in [14], a multi-timescale constrained reinforcement learning framework is proposed to address a constrained optimization problem, and solving the optimization problem is converted into a task of learning the parameters of the neural network and the value of the dual variables. The authors in [19] have proposed a similar approach to address a resource allocation problem in a wireless system. In [20], a learning-based algorithm is proposed for the UAVs to adjust their altitudes and channel access strategies to maximize the total network capacity while satisfying energy constraints. The application of DRL in CDMP problems in the context of radar resource management has not been adequately addressed yet. Additionally, conventional DRL algorithms may not be directly applicable in a radar environment due to the scarcity of radar return signals, the high maneuverability of the targets, and the rapid variations in the environment, motivating the work in this paper.

B. Contributions

In this work, we first formulate the resource allocation problem in multi-target tracking as a CMDP. We have the goal to minimize the variance of the estimations in tracking while complying with the budget constraint. Following the CMDP formulation, we propose a constrained deep reinforcement learning (CDRL) framework to find a near-optimal budget allocation strategy in the considered problem, in which the parameters of the DQN and the dual variables are learned simultaneously.

Training data for DRL algorithms can be scarce and a DRL algorithm with insufficient training data can easily fail, especially in a fast-varying radar environment. To address this challenge, we periodically employ CDRL in an offline manner, where the algorithm evolves in a virtual environment generated based on the current estimations and prior statistical knowledge on the problem. The proposed offline CDRL can provide more data to the algorithm and we demonstrate that employing offline CDRL periodically can stabilize the performance of the proposed CDRL algorithm. We also show that the proposed CDRL framework is robust to sudden changes in the environment. It is also scalable and can adapt its strategy when there is a new target joining the environment.

The remainder of the paper is organized as follows. In Section II, we introduce the radar target tracking model and briefly discuss the operation of the extended Kalman filter. In Section III, we formulate the problem of time allocation in multi-target radar tracking as a constrained Markov decision process. In Section IV, we address this problem via deep reinforcement learning and develop an online CDRL algorithm. We further design an offline algorithm in Section V to complement the online algorithm and develop a hybrid CDRL framework. We present the simulation results in Section VI and conclude the paper in Section VII.

II. Radar Tracking Model

We consider a 2D multi-target tracking problem with extended Kalman filter. In this section, details of the radar tracking model are discussed first and then the application of the extended Kalman filter is illustrated.

A. Target Motion Model

In time slot t, the current state of a target is described as $\mathbf{x_t} = [x_t, y_t, \dot{x}_t, \dot{y}_t]^T$, where (x_t, y_t) are the coordinates of the current location and $\dot{x}_t$, $\dot{y}_t$ are the current horizontal and vertical velocities of the target. Considering a constant velocity model within the revisit interval, the next state evolves from $\mathbf{x_t}$ to

$$\mathbf{x_{t+1}} = \mathbf{F_t x_t} + \mathbf{w_t}, \tag{1}$$

where $\mathbf{F_t} \in \mathbb{R}^{4\times 4}$ is the transition matrix defined as

$$\mathbf{F_t} = \begin{bmatrix} 1 & 0 & T_t & 0 \\ 0 & 1 & 0 & T_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where T_t is the revisit interval at time t chosen by the radar system for tracking the given target [7]. Above, $\mathbf{w_t}$ denotes the maneuverability noise, which is a multivariate zero-mean Gaussian noise with the covariance matrix

$$\mathbf{Q_t} = \begin{bmatrix} T_t^4/4 & 0 & T_t^3/2 & 0 \\ 0 & T_t^4/4 & 0 & T_t^3/2 \\ T_t^3/2 & 0 & T_t^2 & 0 \\ 0 & T_t^3/2 & 0 & T_t^2 \end{bmatrix} \sigma_w^2 \tag{3}$$

where σ_w is the maneuverability noise variance of the target at time t [7].

B. Measurement Model

It is assumed that radar obtains the measurements of the target's current range r and azimuth angle θ for estimating

its location. Let us denote the measurement vector by $\mathbf{z_t}$ and denote the non-linear function that maps $\mathbf{x_t}$ to $\mathbf{z_t}$ by $h(\cdot)$. Now, we can express

$$\mathbf{z_t} = h(\mathbf{x_t}) + \mathbf{v_t} = \left[\sqrt{x_t^2 + y_t^2}, \quad \tan^{-1}\left(\frac{y_t}{x_t}\right)\right]^T + \mathbf{v_t} \quad (4)$$

where $\mathbf{v_t} = [v_{r,t}, v_{\theta,t}]^T$ denotes the measurement noise vector, which consists of the range measurement noise ($v_{r,t}$) and the angle measurement noise ($v_{\theta,t}$) at time t. $v_{r,t}$ and $v_{\theta,t}$ are zero-mean Gaussian noise components with variances $\sigma_{r,t}^2$ and $\sigma_{\theta,t}^2$, respectively. Without loss of generality, it is assumed that the radar is located at the origin of the Cartesian coordinate system.

The variance of the measurement noise is dynamic and depends on the signal-to-noise ratio (SNR_t) of the echo radar signal reflected from the target in time slot t. In this work, it is assumed that the SNR_t only depends on the dwell time τ_t of the radar system and the distance r_t between the target and the radar. SNR_t can be formulated as [7], [21]

$$\text{SNR}_t(\tau_t, r_t) = \text{SNR}_0 \left(\frac{\tau_t}{\tau_0}\right)\left(\frac{r_t}{r_0}\right)^{-4} \quad (5)$$

where SNR_0, τ_0 and r_0 are the reference values of the SNR, dwell time and the distance from target to radar. Then, the relationship between the variance of measurement noise and the SNR can be determined as [22]

$$\sigma_{\bullet,t}^2 = \frac{\sigma_{\bullet,0}^2}{\text{SNR}_t(\tau_t, r_t)} \quad (6)$$

where $\bullet \in (r, \theta)$. $\sigma_{\bullet,0}^2$ denotes the reference value of the corresponding measurement noise variance whose values will be introduced later in the simulation settings. Note that the variance of the measurement noise for a target decreases when longer dwell time is allocated to this target or when the target moves closer to the radar.

Note also that the mapping function $h(\cdot)$ between measurements and states is non-linear and hence extended Kalman filter (EKF) is employed in this work. When using EKF, an observation matrix $\mathbf{H_t} \in \mathbb{R}^{2\times 4}$ is introduced to linearize the relationship between $\mathbf{z_t}$ and $\mathbf{x_t}$. $\mathbf{H_t}$ is defined as the Jacobian of the measurement function $h(\cdot)$:

$$\mathbf{H_t} = \frac{\partial h(\cdot)}{\partial \mathbf{x}}|_{\mathbf{x_t}} = \begin{bmatrix} \frac{x_t}{\sqrt{x_t^2+y_t^2}} & \frac{y_t}{\sqrt{x_t^2+y_t^2}} & 0 & 0 \\ \frac{-y_t}{x_t^2+y_t^2} & \frac{x_t}{x_t^2+y_t^2} & 0 & 0 \end{bmatrix}. \quad (7)$$

Considering independent measurements, the covariance matrix of measurements is given by

$$\mathbf{R_t} = \begin{bmatrix} \sigma_{r,t}^2 & 0 \\ 0 & \sigma_{\theta,t} \end{bmatrix}. \quad (8)$$

C. Extended Kalman Filter

Kalman filter is a well-known algorithm for estimating the state of a process, using a series of measurements over time [23]. Extended Kalman filter is the non-linear version of Kalman filter, which can be employed for target tracking in radar systems with non-linear measurements.

Extended Kalman filter consists of two phases, namely predict and update. For a target-tracking problem, a simplified version of the procedure is as follows:

1) Predict:

$$\hat{\mathbf{x}}_{\mathbf{t|t-1}} = \mathbf{F_t}\mathbf{x}_{\mathbf{t-1|t-1}} \quad (9)$$

$$\hat{\mathbf{P}}_{\mathbf{t|t-1}} = \mathbf{F_t}\mathbf{P}_{\mathbf{t-1|t-1}}\mathbf{F_t}^T + \mathbf{Q_t} \quad (10)$$

2) Update:

$$\mathbf{K_t} = \hat{\mathbf{P}}_{\mathbf{t|t-1}}\mathbf{H_t}^T(\mathbf{H_t}\hat{\mathbf{P}}_{\mathbf{t|t-1}}\mathbf{H_t}^T + \mathbf{R_t})^{-1} \quad (11)$$

$$\mathbf{x_{t|t}} = \hat{\mathbf{x}}_{\mathbf{t|t-1}} + \mathbf{K_t}(\mathbf{z_t} - h(\hat{\mathbf{x}}_{\mathbf{t|t-1}})) \quad (12)$$

$$\mathbf{P_{t|t}} = (\mathbf{I} - \mathbf{K_t}\mathbf{H_t})\hat{\mathbf{P}}_{\mathbf{t|t-1}} \quad (13)$$

where $\hat{\mathbf{x}}_{\mathbf{t|t-1}}$ and $\hat{\mathbf{P}}_{\mathbf{t|t-1}}$ are the prior estimate and the corresponding covariance matrix. $\mathbf{x_{t|t}}$ and $\mathbf{P_{t|t}}$ are the posterior estimate and the corresponding covariance matrix.

In the prediction phase, we use (9) to compute the prior state estimate $\hat{\mathbf{x}}_{\mathbf{t|t-1}}$ with its covariance matrix defined in (10). In the update phase, the optimal Kalman gain $\mathbf{K_t}$ is defined as in (11), which is a weight given to the measurements and current estimation. (12) can be utilized to fuse the received measurements with the current estimation and obtain the posterior state estimate $\mathbf{x_{t|t}}$. And the posterior covariance matrix $\mathbf{P_{t|t}}$ is computed via (13).

In this work, $\mathbf{x_{t|t}}$ is initialized as a zero vector and $\mathbf{P_{t|t}}$ is initialized as an identity matrix, assuming that the system has no prior knowledge of the targets. Extended Kalman filter recursively refines $\mathbf{x_{t|t}}$ by minimizing the trace of the posterior estimate covariance matrix $\mathbf{P_{t|t}}$.

D. Tracking Cost Function

We consider a similar cost function as defined in [7]. Specifically, the tracking cost function at time t is defined as

$$c_t(T_t, \tau_t) = \text{trace}(\mathbf{E}\mathbf{P_{t|t}}\mathbf{E}^T) \quad (14)$$

where

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (15)$$

According to (14), the tracking cost is a function of both the revisit interval (T_t) and the dwell time (τ_t) and can be interpreted as the sum of the posterior estimate variances of the target's current estimated x-axis and y-axis position.

III. Problem Formulation

A. Constrained Markov Decision Processes (CMDP)

A constrained Markov decision process [24] is defined by the tuple $(S, A, C, \Theta, T, \mu, \gamma)$. Here, S denotes the set of states, A is the set of available actions, $C : S \times A \times S \to \mathbb{R}$ is the cost function, $\Theta : S \times A \times S \to \mathbb{R}$ is the budget, $T : S \times A \times S \to [0, 1]$ is the transition probability, μ is the distribution of the initial state. $\gamma \in [0, 1)$ is the discount factor for future rewards and costs. In this paper, we consider finding an optimal policy $\pi : S \times A \to [0, 1]$, which maps the states to the actions, to minimize the discounted sum of the future

costs, while constraining the discounted sum budget violation, i.e.,

$$\begin{aligned} \min_{\pi} \quad & \sum_{m=0}^{\infty} \gamma^m c_{t+m} \\ \text{s.t.} \quad & \sum_{m=0}^{\infty} \gamma^m (\Theta_{t+m} - \Theta_{max}) \leq 0 \end{aligned} \tag{16}$$

where c_t is the tracking cost defined in (14) and Θ_{max} is the total budget constraint. More details and generalization of the above optimization to multi-target tracking are discussed next.

B. Multi-target Tracking with Budget Constraint

We introduced EKF for tracking a single target in Section II. Since our objective is to address multi-target tracking, we consider in the remainder of the paper a radar system for tracking N targets. To synchronize the tracking process for all the targets, it is assumed that the revisit interval for all the targets is the same, i.e. for target n during time t, the selected revisit interval will be $T_t^n = T_0$. Hence, the primary focus is on the allocation of the dwell time τ_t^n for different targets.

Following [14] and [24], the problem considered in this paper can be formulated as the following constrained optimization problem:

$$\begin{aligned} \max_{\pi} \quad & \sum_{m=0}^{\infty} \left[-\gamma^m \sum_{n=1}^{N} c_{t+m}^n(\tau_{t+m}^n) \right] \\ \text{s.t.} \quad & \sum_{m=0}^{\infty} \gamma^m \left(\sum_{n=1}^{N} \frac{\tau_{t+m}^n}{T_0} - \Theta_{max} \right) \leq 0. \end{aligned} \tag{17}$$

Since a multi-target scenario is considered, we denote the tracking cost for target n at time t as $c_t^n(\tau_t^n)$, which is a function of the dwell time τ_t^n allocated to target n at time t and is defined in (14).

The problem can be treated as a time-slotted system with the assumption of fixed revisit interval T_0. Each time slot will last T_0 seconds starting from the beginning of the current radar pulse and ending before sending the next pulse for the same target. It is assumed that the measurements are taken independently and we have $\tau_t^n \in [0, T_0]$ for all possible t and n.

Budget for target n during time slot t is defined as the ratio of the selected dwell time τ_t^n and the fixed revisit interval T_0, i.e., $\Theta_t = \frac{\tau_t^n}{T_0}$. $\Theta_{max} \in [0, 1]$ is the total budget constraint for tracking all N targets in the radar system.

As shown in the optimization problem (17), the goal of this work is to find a budget (or equivalently dwell time) allocation policy π for minimizing the discounted sum of the future total tracking costs of all the targets while constraining the discounted sum budget violation. In this work, we employ deep reinforcement learning, and hence policy π is determined by N deep Q-networks (DQNs) of the deep reinforcement learning (DRL) algorithm. Details are provided in the next section.

With Lagrangian relaxation, the budget constraint in (17) can be incorporated into the objective function. By introducing a non-negative dual variable λ_t, problem (17) can be relaxed to the following unconstrained optimization problem:

$$\begin{aligned} \min_{\lambda_t \leq 0} \max_{\pi} \sum_{m=0}^{\infty} \gamma^m \Bigg[& - \sum_{n=1}^{N} c_{t+m}^n(\tau_{t+m}^n) \\ & - \lambda_t \left(\sum_{n=1}^{N} \frac{\tau_{t+m}^n}{T_0} - \Theta_{max} \right) \Bigg]. \end{aligned} \tag{18}$$

The optimization problem in (18) is known as the dual of the prime problem in (17) and the duality gap between the two problems becomes null for convex optimization problems. It is worth noting that λ_t is a time-varying variable and arbitrarily selecting its value will lead to a sub-optimal solution.

In the next section, we propose a constrained DRL (CDRL) framework to address the unconstrained optimization problem in (18). In the proposed framework, values of the DQN parameters and the dual variable are learned simultaneously.

IV. Constrained Deep Reinforcement Learning for Multi-target Tracking

In this section, we introduce the proposed CDRL framework for multi-target tracking in a radar system. The goal is to find a budget allocation policy π such that the total cost for all the targets is minimized, i.e., the total variance of estimating the locations of the targets is minimized, while the total budget is constrained to be under a certain threshold. As discussed next, we utilize deep Q-learning (DQL) to achieve this goal.

A. Deep Q-Learning

In reinforcement learning, agents learn an optimal or near-optimal policy by interacting with the environment and obtaining a reward for the action taken. The objective of the agent is to learn a policy that can maximize the discounted sum reward,

$$R = \sum_{m=0}^{\infty} \gamma^m r_{t+m} \tag{19}$$

where γ is the discount factor and r_t is the instantaneous reward at time t.

In Q-learning, $Q^{\pi}(\mathbf{s_t}, a_t)$ is defined as the action-value function, which quantifies the expected discounted future sum reward that can be obtained by taking action a_t in state $\mathbf{s_t}$ and following policy π thereafter. Q values are updated as

$$Q^{\pi}(\mathbf{s_t}, a_t) = Q^{\pi}(\mathbf{s_t}, a_t) + \alpha(r_t + \gamma \max_{a'} Q^{\pi}(\mathbf{s_{t+1}}, a')) \tag{20}$$

where $\mathbf{s_{t+1}}$ is the next state and α is the learning rate.

We assume that the agent selects the actions based on the ϵ-greedy method. In this scheme, the agent chooses the action with the largest $Q^{\pi}(\mathbf{s_t}, a_t)$ value with probability $1 - \epsilon$ and selects an action randomly with probability ϵ.

Classical tabular Q-learning stores Q values in a Q-table, and functions well with small state and action spaces. As the size of the state space or the number of available actions increases, deep Q-learning starts to outperform the tabular Q-learning. In deep Q-learning, a deep Q network (DQN) is used

to map the state-action pairs to the corresponding Q-values. Parameters θ^{π} of the DQN are updated by minimizing the following loss function L via experience replay and back-propagation:

$$L(\theta^{\pi}) = (Q^{\pi}(\mathbf{s_t}, a_t) - (r_t + \gamma \max_{a'} Q^{\pi}(\mathbf{s_{t+1}}, a')))^2. \quad (21)$$

In this work, we employ DRL with multiple agents. Specifically, it is assumed that there exist N agents or equivalently N DQNs in the radar system, and each is assigned to tracking one of the N targets. Each DQN autonomously selects its action a_t^n based on its current state $\mathbf{s_t^n}$. The DQN parameters of agent n are denoted as $\theta^{\pi,n}$.

B. Proposed CDRL Framework

1) State: State $\mathbf{s_t^n}$ denotes the current state of agent n at time t, which consists of partial observations of the underlying MDP. In this work, $\mathbf{s_t^n}$ is defined as

$$\mathbf{s_t^n} = [x_{n,t-1}, y_{n,t-1}, \dot{x}_{n,t-1}, \dot{y}_{n,t-1}, d_{n,t-1}, \bar{d}_{t-1}] \quad (22)$$

where $x_{n,t-1}$, $y_{n,t-1}$, $\dot{x}_{n,t-1}$ and $\dot{y}_{n,t-1}$ are the posterior estimated location and velocity of target n in the previous time slot $t-1$ (estimated using the radar returns and extended Kalman filter (EKF) as described in Section II-C). $d_{n,t-1} = \sqrt{x_{n,t-1}^2 + y_{n,t-1}^2}$ is the estimated distance from radar to target n. $\bar{d}_{t-1} = \frac{\sum_{i \neq n} d_{i,t-1}}{N-1}$ is the average of the estimated distances of the other $N-1$ targets. With the last term $\bar{d}_{t-1}$, agent n has access to limited knowledge on the other targets.

2) Action: Based on the current state $\mathbf{s_t^n}$, each agent selects an action a_t^n. a_t^n is defined as the dwell time τ_t^n selected by agent n in time slot t. The possible values of the dwell time are within the range $\tau_t^n \in (0, T_0]$. This range is quantized to L levels and hence the size of the action space is L.

3) Reward: Given the value of λ_t, the reward function r_t is defined as

$$r_t = -\sum_{n=1}^{N} c_t^n(\tau_t^n) - \lambda_t \left(\sum_{n=1}^{N} \frac{\tau_t^n}{T_0} - \Theta_{max} \right) \quad (23)$$

where $c_t^n(\tau_t^n)$ is the tracking cost of target n in time slot t.

The reward r_t is shared among all N agents located at the radar and hence the agents collaboratively maximize this global objective function.

The value of λ_t is critical and an arbitrary value may lead to a sub-optimal solution for the optimization problem. In the remainder of this section, we introduce how to simultaneously learn the DQN parameters $\theta^{\pi,n}$ and the time-varying dual variable λ_t.

C. Online CDRL Algorithm

The proposed CDRL algorithm can be divided into two parts: update of the DQNs $\{\theta^{\pi,n}\}_{n=1}^N$ and update of the dual variable λ_t. The structure of the algorithm is depicted in Fig. 1.

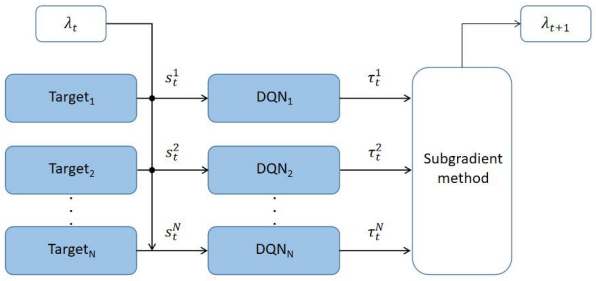

Fig. 1. CDRL Framework

1) Update of the DQNs: At the beginning of time slot t, agent n will have access to its current state $\mathbf{s_t^n}$, which consists of the posterior estimates obtained in the previous time slot. Each agent (for any n) then selects its action a_t^n based on $\mathbf{s_t^n}$ using its DQN, i.e. $a_t^n = \pi^{\theta^{\pi,n}}(\mathbf{s_t^n})$, where $\pi^{\theta^{\pi,n}}$ denotes the DQN with the parameters $\theta^{\pi,n}$. Once agent n selects an action, the cost $c_t^n(\tau_t^n)$ and the next state $\mathbf{s_{t+1}^n}$ can be calculated with the EKF as discussed in Section II.

λ_t is a constant during the entire time slot t and it will only be updated at the end of the time slot. Given λ_t, the reward function r_t is defined as in (23) and the experience tuple ($\mathbf{s_t^n}$, a_t^n, r_t, $\mathbf{s_{t+1}^n}$) will be stored into the experience replay buffer for training the DQN.

The procedures are the same for all the N agents. By setting the reward function as (23), the agents will learn to collaboratively maximize the discounted reward r_t:

$$\max_{\{\theta^{\pi,n}\}_{n=1}^N} \sum_{m=0}^{\infty} \gamma^m \Bigg[-\sum_{n=1}^{N} c_{t+m}^n(\tau_{t+m}^n) - \lambda_t \left(\sum_{n=1}^{N} \frac{\tau_{t+m}^n}{T_0} - \Theta_{max} \right) \Bigg]. \quad (24)$$

From the perspective of agent n, solving the optimization problem in (24) is equivalent to updating its DQN parameters by minimizing the loss function (21)

$$\theta_{t+1}^{\pi,n} = \theta_t^{\pi,n} - \eta \bigtriangledown_{\theta_t^{\pi,n}} L(\theta_t^{\pi,n}) \quad (25)$$

where η denotes the learning rate of the DQN parameters. (25) can be handled via the method of experience replay and backpropagation in the DRL algorithm.

2) Update of the Dual Variable: At the end of time slot t, each agent has determined its action $a_t^n = \tau_t^n$ with the assistance of its DQN. We denote the objective function in (18) as $\mathcal{L}$. Then, λ_t is updated by minimizing $\mathcal{L}$ over λ_t, i.e.

$$\lambda_{t+1} = \max(0,\ \lambda_t - \alpha \bigtriangledown_{\lambda_t} \mathcal{L}) = \max\left(0,\ \lambda_t + \alpha \sum_{m=0}^{\infty} \gamma^m \left(\sum_{n=1}^{N} \frac{\tau_{t+m}^n}{T_0} - \Theta_{max} \right) \right) \quad (26)$$

where α is the learning rate of the dual variable. The gradient $\bigtriangledown_{\lambda_t} \mathcal{L}$ can be estimated with an additional neural network but

we simplify instead as

$$\lambda_{t+1} = \max\left(0,\ \lambda_t + \alpha\left(\sum_{n=1}^{N}\frac{\tau_t^n}{T_0} - \Theta_{max}\right)\right). \quad (27)$$

Note that the dual variable λ_t increases when the total budget exceeds the threshold, which further increases the violation penalty added to r_t. On the other hand, if the total budget is always below the threshold, λ_t will keep decreasing to zero and problem (17) becomes an unconstrained optimization problem. The value of λ_t varies along with the evolving dynamic environment. Details of the proposed CDRL framework can be found in Algorithm 1 below.

The proposed CDRL algorithm iteratively determined the DQN parameters and the dual variable (θ_t, λ_t). From the proofs in [14] and also in Chapter 6 of [25], the iterative method of solving (θ_t, λ_t) can be seen as a multi-timescale stochastic approximation process and converges to a fixed point (θ_t^*, λ_t^*).

Algorithm 1 Online CDRL Algorithm

1: Initialize DQNs parameters $\{\theta_0^{\pi,n}\}_{n=1}^N$ with random values.
2: Initialize states $\{\mathbf{s_0^n}\}_{n=1}^N$ as zero vectors and λ_t as λ_0.
3: **for** time slot $t = 0, 1, ..., T_{max}$ **do**
4: **for** each agent $n = 1, 2, ..., N$ **do**
5: Select an action a_t^n based on the current state $\mathbf{s_t^n}$ with its DQN $\theta_t^{\pi,n}$ and ϵ-greedy method.
6: Obtain $c_t^n(\tau_t^n)$ and $\mathbf{s_{t+1}^n}$ with the measurements $\mathbf{z_t^n}$ via extended Kalman filter.
7: **end for**
8: Compute reward r_t according to (23).
9: **for** each agent $n = 1, 2, ..., N$ **do**
10: Store the experience ($\mathbf{s_t^n}$, a_t, r_t, $\mathbf{s_{t+1}^n}$) to its own DQN experience buffer.
11: Update $\theta_t^{\pi,n}$ to $\theta_{t+1}^{\pi,n}$ with experience replay and back-propagation.
12: **end for**
13: $\lambda_{t+1} = \max(0,\ \lambda_t - \alpha(\sum_{n=1}^N \frac{\tau_t^n}{T_0} - \Theta_{max}))$.
14: **end for**

Algorithm 1 is performed in an online manner. For a multi-target tracking problem in a radar system, each agent can only obtain one training data ($\mathbf{s_t^n}$, a_t, r_t, $\mathbf{s_{t+1}^n}$) every time slot since the cost $c_T^n(\tau_t^n)$ and the estimations $\mathbf{s_{t+1}^n}$ cannot be generated without the measurements $\mathbf{z_t^n}$. Training a neural network usually requires a large quantity of data. DQN algorithm performs experience replay on the memory buffer and reuses the past data to alleviate this issue. However, challenges arise when dealing with a fast-varying environment such as in the multi-target tracking problem in which all the targets keep moving in a given area. In such cases, it is desired to perform sufficiently many updates of the DQNs in a short period of time for adapting to the fast-varying environment. On the other hand, too many updates over a small amount of training data will lead to overfitting and hence degrade the performance.

In the next section, we develop an offline CDRL framework to address this difficulty by generating virtual training data to stabilize the training of DQNs.

V. Offline and Hybrid Constrained Deep Reinforcement Learning for Multi-target Tracking

It is challenging to get access to enough training data in the multi-target tracking problem. In the online CDRL algorithm introduced in the previous section, each training data ($\mathbf{s_t^n}$, a_t, r_t, $\mathbf{s_{t+1}^n}$) for agent n requires a set of measurements $\mathbf{z_t^n} = [r_t^n, \theta_t^n]^T$ of target n in time slot t. One can acquire more data by decreasing the revisit interval T_0, but that renders the tracking tasks more costly.

In this section, we propose an offline CDRL framework in which a virtual future of the environment is generated based on the current estimates and hence more training data can be obtained without taking real measurements.

Fig. 2 and Fig. 3 depict the workflow of the online and the offline CDRL algorithms, respectively. Note that in offline CDRL, the agent in between two consecutive real measurements is further is updated by the generated virtual measurements as depicted by the upper branch in Fig. 3. How frequently the agent is updated via virtual measurements is controlled by the switch S. In offline CDRL, switch is always closed. Hence, virtual measurements are generated and updates are done in between each consecutive real measurements. If the switch is on only periodically (to reduce the computational complexity), we have hybrid CDRL as will described in more detail towards the end of the section.

In this section, we focus on the following variables that are critical to the learning algorithm: states $\{\mathbf{s_t^n}\}_{n=1}^N$, their corresponding covariance matrices $\{\mathbf{P_{t|t}^n}\}_{n=1}^N$ and the dual variable λ_t. After receiving the measurements $\mathbf{z_t^n}$, online CDRL algorithm will perform the updates and compute the reward r_t as shown in Fig. 2.

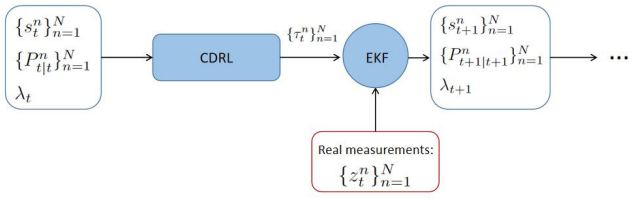

Fig. 2. Block diagram of online CDRL

A. Generation of Virtual Future

For obtaining more training data, we generate multiple episodes of virtual future within the environment and the horizon of each episode is H. Subsequently, the CDRL framework will be trained on this generated environment instead of the real one, in an offline manner. Note that the goal is to prepare the DQN well enough before employing it in the real environment.

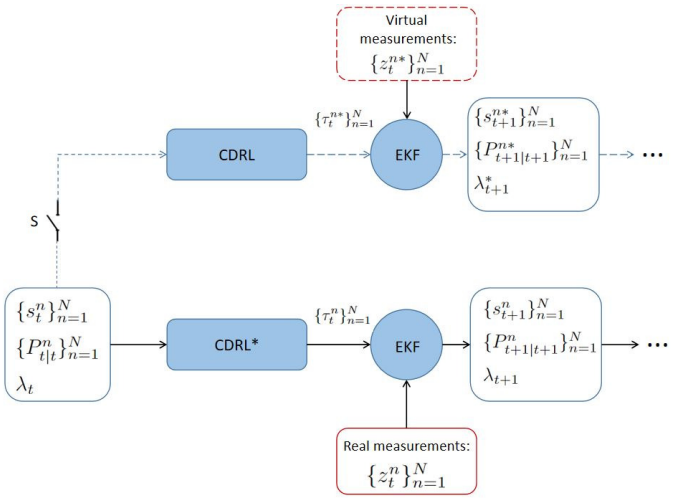

Fig. 3. Block diagram of offline/hybrid CDRL

The virtual future is generated after obtaining $\{\mathbf{s_t^n}\}_{n=1}^N$, $\{\mathbf{P_{t|t}^n}\}_{n=1}^N$, λ_t and before receiving the upcoming measurements $\mathbf{z_t^n}$. The upper path in Fig. 3 demonstrates one possible episode of the virtual future. Each episode starts from the same copy of the current $\{\mathbf{s_t^n}\}_{n=1}^N$, $\{\mathbf{P_{t|t}^n}\}_{n=1}^N$ and λ_t and evolves until the horizon H is reached.

For agent n, each episode can be expressed as a trajectory of experience ($\mathbf{s_t^n}$, a_t^*, r_t^*, $\mathbf{s_{t+1}^{n*}}$, a_{t+1}^*, r_{t+1}^*, ..., $\mathbf{s_{t+H}^{n*}}$) and hence H training data can be obtained from one episode of the generated future.

B. Generation of Virtual Measurements

Different from the real measurements which are obtained based on the actual locations of the targets, virtual measurements of target n are generated based on the estimates ($x_{n,t-1}$, $y_{n,t-1}$ $\dot{x}_{n,t-1}$, $\dot{y}_{n,t-1}$) in $\mathbf{s_t^n}$, which is the estimated location and velocity of target n in the previous time slot. Current location of target n can then be estimated as

$$(x_{n,t}^*, y_{n,t}^*) = (x_{n,t-1} + T_0\dot{x}_{n,t-1}, y_{n,t-1} + T_0\dot{y}_{n,t-1}) \quad (28)$$

where $x_{n,t}^*$ and $y_{n,t}^*$ denote the estimated coordinates of target n. And the virtual measurement $\mathbf{z_t^{n*}}$ can be generated based on $(x_{n,t}^*, y_{n,t}^*)$ as

$$\mathbf{z_t^{n*}} = \left[\sqrt{{x_{n,t}^*}^2 + {y_{n,t}^*}^2}, \quad \tan^{-1}\left(\frac{y_{n,t}^*}{x_{n,t}^*}\right)\right]^T + \mathbf{v_t^{n*}} \quad (29)$$

where $\mathbf{v_t^{n*}}$ is the measurement noise of the virtual measurements, which is determined by both the action a_t^* and the estimated distance $d_{n,t}^* = \sqrt{{x_{n,t}^*}^2 + {y_{n,t}^*}^2}$. The virtual measurements will be generated every step during an episode of the generated virtual future.

Algorithm 2 illustrates how to generate an episode of virtual future at time t. Variables denoted with $*$ will only occur during the episodes of virtual future. It can be seen from the algorithm that each of the episodes starts from the same state of the system and evolves in the virtual environment with the virtual measurements.

Algorithm 2 Generating Virtual Future

1: Set $\{\mathbf{s_t^{n*}}\}_{n=1}^N = \{\mathbf{s_t^n}\}_{n=1}^N$, $\{\mathbf{P_{t|t}^{n*}}\}_{n=1}^N = \{\mathbf{P_{t|t}^n}\}_{n=1}^N$, $\lambda_t^* = \lambda_t$.
2: **for** horizon $h = 0, 1, ..., H$ **do**
3: **for** each agent $n = 1, 2, ..., N$ **do**
4: Select an action a_{t+h}^{n*} based on the current state $\mathbf{s_{t+h}^{n*}}$ with its DQN and ϵ-greedy method.
5: Obtain the cost $c_{t+h}^{n*}(\tau_{t+h}^{n*})$ and the next state $\mathbf{s_{t+h+1}^{n*}}$ with the virtual measurements $\mathbf{z_{t+h}^{n*}}$ via extended Kalman filter.
6: **end for**
7: Compute reward r_{t+h}^* according to (23).
8: **for** each agent $n = 1, 2, ..., N$ **do**
9: Store the experience ($\mathbf{s_{t+h}^{n*}}$, a_{t+h}^*, r_{t+h}^*, $\mathbf{s_{t+h+1}^{n*}}$) to its own DQN experience buffer.
10: Update its DQN with experience replay and back-propagation.
11: **end for**
12: $\lambda_{t+h+1}^* = \max(0,\ \lambda_{t+h}^* - \alpha(\sum_{n=1}^N \frac{\tau_{t+h}^{n*}}{T_0} - \Theta_{max}))$.
13: **end for**

Algorithm 3 Offline CDRL Algorithm

1: Initialize DQNs parameters $\{\theta_0^{\pi,n}\}_{n=1}^N$ with random values.
2: Initialize states $\{\mathbf{s_0^n}\}_{n=1}^N$ as zero vectors and λ_t as λ_0.
3: **for** time slot $t = 0, 1, ..., T_{max}$ **do**
4: **for** episode $e = 0, 1, ..., E_{max}$ **do**
5: Call the function **Generating Virtual Future**.
6: **end for**
7: **for** each agent $n = 1, 2, ..., N$ **do**
8: Select an action a_t^n based on the current state $\mathbf{s_t^n}$ with its DQN.
9: Obtain $c_t^n(\tau_t^n)$ and $\mathbf{s_{t+1}^n}$ with the measurements $\mathbf{z_t^n}$ via extended Kalman filter.
10: **end for**
11: **end for**
12: $\lambda_{t+1} = \max(0,\ \lambda_t - \alpha(\sum_{n=1}^N \frac{\tau_t^n}{T_0} - \Theta_{max}))$.

The proposed offline CDRL framework is summarized in Algorithm 3. In each time slot, the DQNs will first be trained with E_{max} episodes of generated future as shown in steps 4 through 6. Then, the DQNs will be used for the actual decision making, as shown in steps 7 through 10. The strength of offline CDRL is that it generates more training data without taking real measurements, which helps to improve the performance of CDRL in a fast-varying environment. However, employing CDRL in a purely offline manner can also become problematic. It is worth noting that the virtual future is generated based on the estimated current state of the targets and some prior knowledge of the environment, such as the reference values of the measurement and maneuverability noise variance. In practice, these values can vary depending on the changes in

the environment. For instance, the variance of measurement noise can increase when the corresponding target moves into a region, for instance, with poor weather conditions. In such cases, utilizing online data/observation becomes critical for adapting to the change in the environment.

Considering this, we propose a *hybrid CDRL* framework which combines online and offline CDRL, where offline CDRL is performed periodically every T_{offline} time slots and online CDRL is performed in the remainder of the time. The workflow of offline/hybrid CDRL algorithm is depicted in Fig. 3, where the switch S, as described before, controls the generation of the virtual future. For purely offline CDRL, the switch in Fig. 3 is always closed and the virtual future is generated in between each consecutive real measurements. For hybrid CDRL, the switch is closed every T_{offline} time slots to reduce the computational burden. When the switch is closed, the DQN is updated to a newer version with the virtual data generated with the upper branch in Fig. 3 and CDRL* will copy the updated version of the DQN. In this way, the experience replay buffer is a mixture of both the online and offline training data. The offline component provides more training data to the algorithm while the online learning enables the algorithm to adapt to the dynamic environment.

VI. Simulation Results

A. Simulation Setup

TABLE I
Simulation Parameters

$\sigma^2_{r,0}$ (m^2)	100
$\sigma^2_{\theta,0}$ (rad^2)	4e-4
σ_w $((m/s^2)^2)$	25
Reference distance r_0 (m)	5000
Reference dwell time τ_0 (s)	1
Revisit interval T_0 (s)	1
Number of episodes of virtual future $E_{\max}$	10
Length of each episode H	10
Period of Performing offline CDRL T_{OFF} (s)	50
DRL discount factor γ	0.9
DRL mini-batch size	16
Exploring probability ϵ	0.05
Initial dual variable (λ_0)	1000
Step size of dual variable (α)	2000

The parameters of the simulations are listed in Table I. We assume that the reference variances of the measurement noise and the maneuverability noise are the same for all the targets, and they are denoted as $\sigma^2_{r,0}$, $\sigma^2_{\theta,0}$ and σ^2_w, respectively.

In the DQN structure, there is a feed-forward layer with 50 neurons, followed by a dueling network consisting of an advantage layer and a value layer with 10 neurons in each. Additionally, there is an input layer for taking states as the input and an output layer for outputting the Q values of the actions. The learning rates of DQNs are set to be 0.005 for offline CDRL and 0.01 for online CDRL.

The size of the experience replay buffer is selected to be 300. We avoid using a large buffer since previous data can easily become outdated in a fast-varying environment. In this section, the simulation results are obtained by averaging 5 runs on each test case.

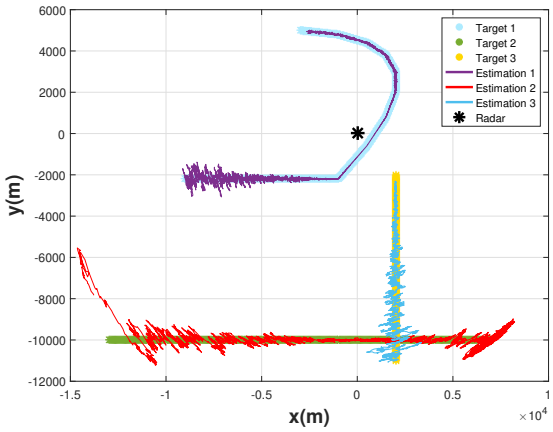

Fig. 4. Case I: Targets' Trajectories and EKF Estimations

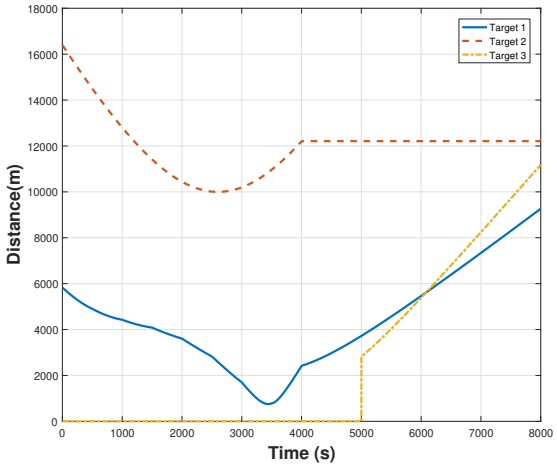

Fig. 5. Case I: Targets' Distances to Radar

B. Test Case I

We assume that the targets are moving in a 2-D region. The trajectories of the targets in Test Case I, along with the EKF estimations, are depicted in Fig. 4. In this first case, the initial position and horizontal and vertical velocities of Target 1 are $(x_{n=1,t=0}, y_{n=1,t=0}) = (-3000m, 5000m)$ and $(\dot{x}_{n=1,t=0}, \dot{y}_{n=1,t=0}) = (5m/s, 0m/s)$ respectively. Every 500 seconds, Target 1 reduces its horizontal (x-axis) velocity by $1m/s$ and reduces its vertical (y-axis) velocity by $0.4m/s$. Target 2 starts from $(-13000m, -10000m)$ and it has the

horizontal and vertical velocities of $(5m/s, 0m/s)$. At $t = 4000s$, Target 1 will change its velocity to $(-2m/s, 0m/s)$ and Target 2 will stop moving. At $t = 5000s$, there is a third target joining the environment. Target 3 starts from $(2000m, -2000m)$ and moves with the horizontal and vertical velocities of $(0m/s, -3m/s)$. Fig. 5 plots the distance to radar for each target. Initially, we set the budget constraint as $\Theta_{max} = 1$. After $t = 3000s$, the budget constraint is reduced to 0.8.

Estimated trajectories by EKF are also plotted in Fig. 4. We note that the estimations are more accurate (as indicated by smaller variations) when the locations of the targets are closer to the radar. This is expected because the variance of the measurement noise diminishes as the distance to the radar decreases, as can be seen in (6).

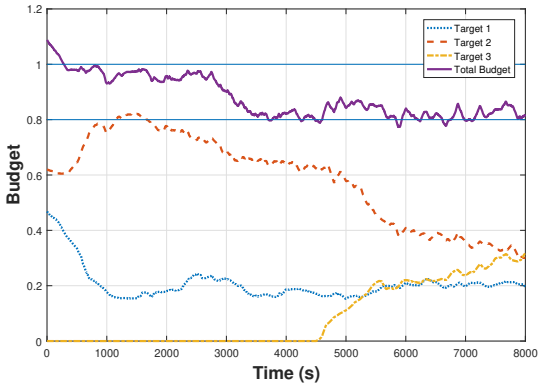

Fig. 6. Case I: Budget Allocation with Hybrid CDRL

Fig. 6 depicts the moving average of the budget allocation strategy learned by hybrid CDRL. When the reward function is defined as in (23), the proposed hybrid CDRL learns a policy that dynamically allocates the budget to different targets while aiming to satisfy the constraint. We notice that more budget (or equivalently larger dwell time) is typically provided for tracking the target which is further away from the radar. Note that a larger dwell time τ_t^n will improve the SNR of the received echo signal and help support the tracking of a target at a larger distance. For instance, we observe in Fig. 5 that Target 2's distance is larger (compared to Target 1's distance) and correspondingly Target 2 is allocated larger dwell time as seen in Fig. 6.

At $t = 3000s$, the budget constraint is reduced from 1 to 0.8 and the proposed algorithm can handle this change and quickly learn a new strategy that satisfies the updated constraint. Indeed, the solid-lined total budget curve deviates from the horizontal straight line at 1 and approaches the horizontal straight line at 0.8, which represents the new budget constraint. This capability of adapting to a new constraint makes it more flexible for the radar system to allocate the rest of the budget to other tasks (e.g. scanning and detecting new targets).

At $t = 5000s$, Target 3 joins and a new agent with an initialized DQN is assigned for tracking this target. When the new target joins, it takes some time for the new agent to train its DQN and also for the existing agents to adapt to the new environment. Due to this, a slight violation of the budget constraint can be observed in Fig. 6 around $t = 5000s$, but the agents can eventually learn a new strategy that achieves close to the budget constraint despite the increased number of agents, which demonstrates the adaptive capability of the proposed hybrid CDRL framework. We further observe by comparing Figs. 5 and 6 that Target 3 moves away from the radar and as a result, the budget allocated to tracking Target 3 increases over time.

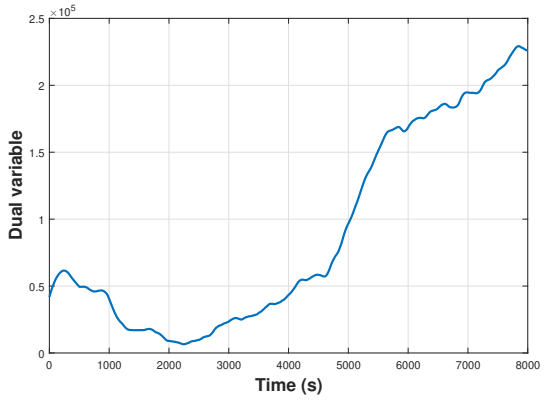

Fig. 7. Case I: Dual Variable

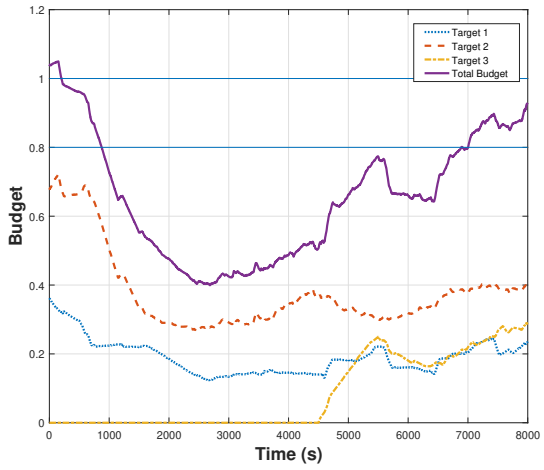

Fig. 8. Case I: Budget Allocation with fixed λ_t

The total budget is constrained via the dual variable λ_t and moving average of the dual variable is plotted in Fig. 7. λ_t increases when there is a violation of the budget constraint and decreases when the constraint is satisfied. λ_t is updated throughout the entire algorithm, and a fixed value of λ_t will lead to a sub-optimal solution. As an example, we arbitrarily set the dual variable to a fixed value of 150000 and the corresponding budget allocation can be seen in Fig. 8. We observe that the total budget is underutilized before $t = 7000s$ because the selected dual variable is larger than the optimal

value during this time. After $t = 7000s$, the budget constraint of 0.8 is significantly violated. Therefore, it is critical to learn the optimal value of the dual variable throughout the entire process.

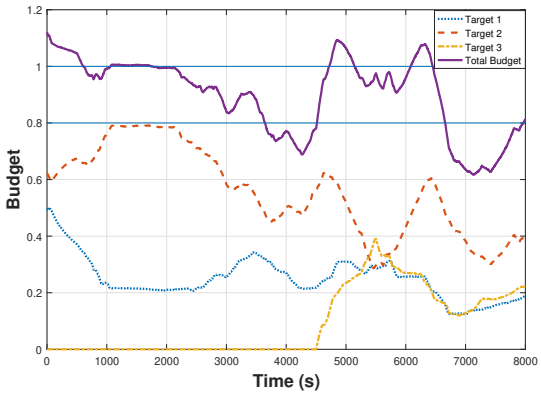

Fig. 9. Case I: Budget Allocation with Online CDRL

Next, we perform an experiment with online CDRL in the same test case and the learned budget allocation strategy is depicted in Fig. 9. Online CDRL demonstrates a relatively strong performance when dealing with a simple task before approximately $t = 3000s$. Overall, online CDRL can learn to allocate more budget to a target with a larger distance but it is difficult for online CDRL to constrain the total budget at the thresholds when the task becomes more complex (e.g., when the budget constraint is reduced and the third target is generated). For instance, we notice that the online CDRL agent initially attempts to satisfy the budget constraint of 0.8, as seen in the diminishing trend of the solid-lined curve in Fig. 9. However, around the time Target 3 joins, there is a substantial violation of this budget. A reason for this observation is that online CDRL does not obtain enough training data from the environment and hence fails to achieve performance as stable as that of hybrid CDRL.

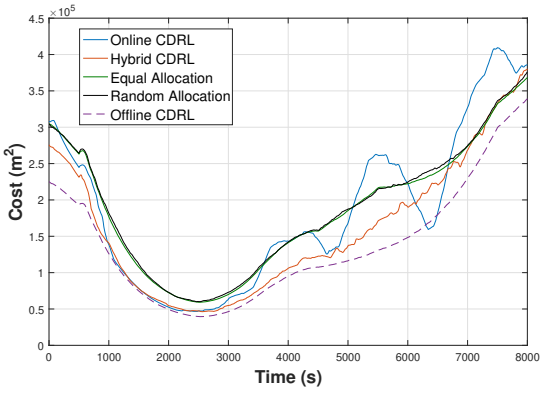

Fig. 10. Case I: Cost Comparison

Fig. 10 compares the total cost achieved by different algorithms. The total cost is defined as the total error variance in estimating the locations of the targets, i.e., it is the sum of the tracking costs defined in (14) summed up over all targets. Overall, hybrid CDRL achieves high performance with consistently lower cost compared to the benchmarks (only fully offline CDRL achieves lower cost as discussed in detail below). It takes a longer time for online CDRL to converge compared to hybrid CDRL, which can be observed in the figure before $t = 1000s$. Subsequently, online CDRL can achieve a performance comparable to that of hybrid CDRL before $t = 3000s$. Later, as the budget constraint is reduced and the new target joins in, online CDRL experiences performance degradation in handling the more complex task, and achieves a performance with a higher total cost. The simulation results indicate that the offline CDRL component is highly critical in providing training data to support the learning algorithm.

In Fig. 10, we also show the results when we employ CDRL in a fully offline manner ($T_{\text{offline}} = 1s$), which achieves the optimal performance among all the algorithms. This is expected since offline CDRL generates a series of training data every time the radar receives a reflected radar echo signal. Such an abundance of training data helps the learning process of the proposed CDRL algorithm. However, offline CDRL can significantly increase the computational complexity of the algorithm. Besides, in Test Case II below, we see that offline CDRL fails to make the necessary adaptations to the changes in the environment due to the lack of sensing the actual environment via the reward mechanism.

We consider two other benchmarks, equal allocation and random allocation. It can be immediately noticed in Fig. 10 that equally allocating the available budget to each target is not efficient in most of the scenarios. It achieves a comparable performance to online CDRL after $t = 7000s$ since equal allocation is desired in this specific case in which the targets have similar distances to the radar. Finally, random allocation demonstrates a performance similar to equal allocation. This is due to the fact that in each decision epoch, a random allocation is decided, and over a period of time, averaging occurs leading to results akin to equal allocation.

C. Test Case II

In the second test case, we consider a scenario with two targets. Target 1 starts from the location $(-3000m, 5000m)$ and the initial horizontal and vertical velocities are $(7m/s, 0m/s)$. Every 500 seconds, Target 1 reduces its horizontal velocity by $1m/s$ and reduces its vertical velocity by $0.4m/s$. Target 2 starts from $(-13000m, -10000m)$ with velocities $(5m/s, 3m/s)$. The true trajectories along with the estimated trajectories are plotted in Fig. 11.

As depicted in Fig. 11, we assume that there is a (rectangular) hazardous region in which the performance of tracking a target is negatively impacted, e.g., due to extremely poor weather conditions. Hence, a minimum budget is required for receiving reliable measurements of the targets in this region.

We further assume that the online CDRL algorithm can sense this negative impact on target tracking via a flag named "DANGER". Specifically, if a target moves into the hazardous area and the budget assigned to this target is below the

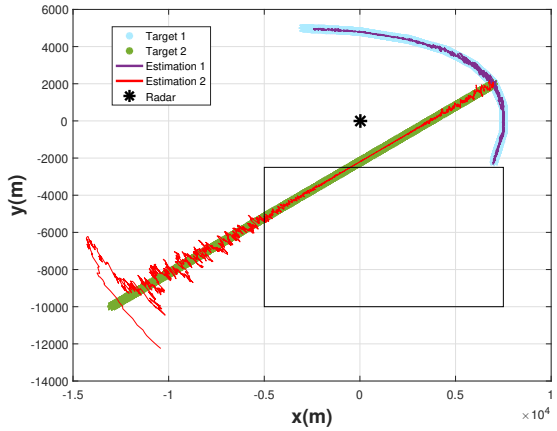

Fig. 11. Case II: Targets' Trajectories and EKF Estimations

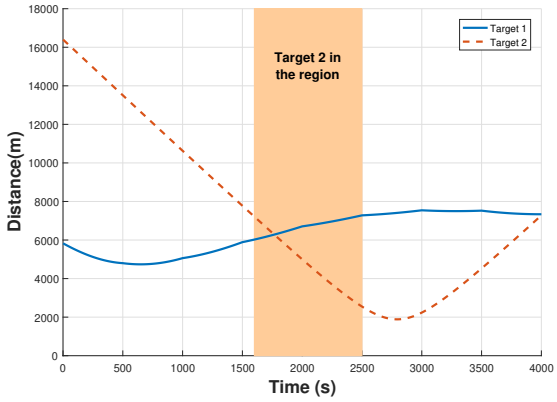

Fig. 12. Case II: Targets' Distances to Radar

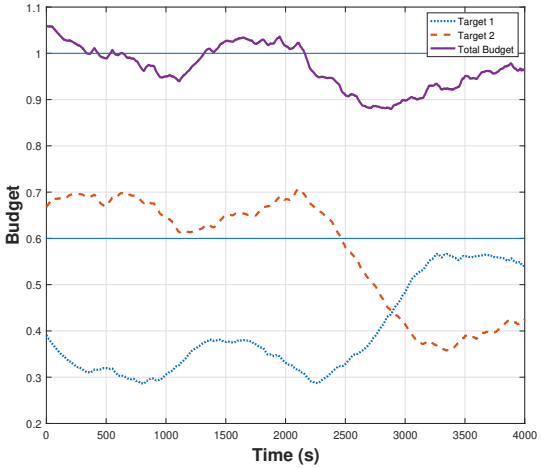

Fig. 13. Case II: Budget Allocation with Hybrid CDRL

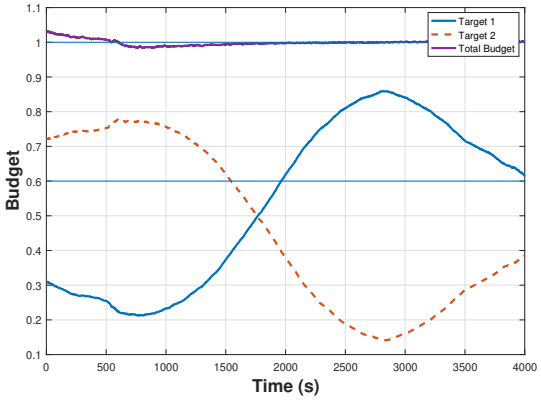

Fig. 14. Case II: Budget Allocation with Offline CDRL

minimum requirement $\Theta_{\text{danger}} = 0.6$, "DANGER" will be set to 1 and a large penalty $P_{\text{danger}} = 4 \times 10^5$ will be added to the cost of the corresponding target. Otherwise, DANGER $= 0$ and there will be no additional penalty.

Fig. 12 shows the distances of the two targets to the radar. Target 2 moves into the hazardous region at $t = 1600s$ and leaves at $t = 2500s$, which can be seen in both Fig. 11 and Fig. 12. It is desired to find a strategy that can allocate the minimum required budget to the target in the hazardous zone.

The budget allocation strategy learned by hybrid CDRL is shown in Fig. 13. Overall, hybrid CDRL is able to constrain the total budget close to the threshold $\Theta_{\max} = 1$. Before $t = 1600s$, the proposed algorithm learns a strategy to allocate more budget to the detection of a distant target, which is similar to Test Case I. When Target 2 moves into the hazardous region at around $t = 1600s$, hybrid CDRL can sense the change in the environment with its online learning component and continue to allocate more budget (above Θ_{danger}) to Target 2 despite the fact that Target 1 has a larger distance to the radar during this time period. After Target 2 leaves the region at $t = 2500s$, hybrid CDRL can adapt to this change and allocate budgets based on the distances of the targets.

Offline CDRL learns a strategy merely based on training data in the virtual environment, which is generated based on the initial knowledge of the environment. We test offline CDRL in Test Case II, and its performance on budget allocation is shown in Fig. 14. It can be seen that agents in offline CDRL tend to allocate budgets based on distances, similar to the strategy in Test Case I. However, it cannot adapt to the change in the environment due to the lack of sensing of the real environment.

The budget allocation strategy of online CDRL in Test Case II is illustrated in Fig. 15. We see that employing CDRL in a purely online manner does not lead to strong performance with respect to constraining the total budget. Besides, this approach leads to the learning of an undesired strategy after $t = 2500s$, when Target 2 leaves the hazardous region. Specifically, after leaving this region, Target 2 has a smaller distance to the radar and should be allocated smaller budget. However, we notice in Fig. 15 that the online CDRL agent keeps allocating more time to the tracking of Target 2.

Costs of different algorithms in Test Case II are compared

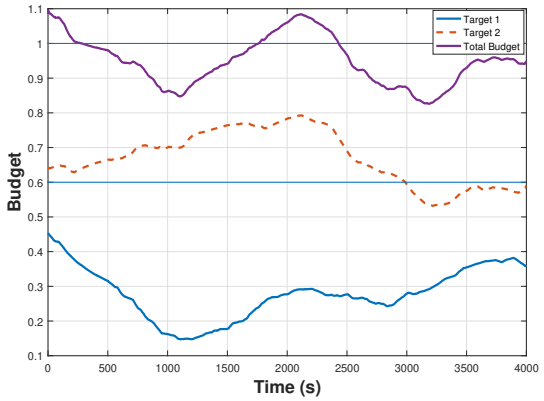

Fig. 15. Case II: Budget Allocation with Online CDRL

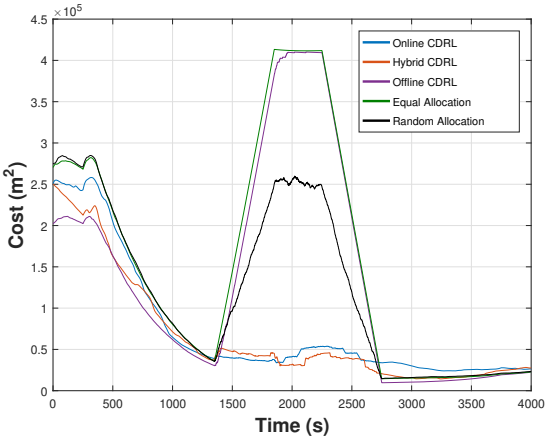

Fig. 16. Case II: Cost Comparison

in Fig. 16. Before Target 2 moves into the hazardous region at $t = 1600s$, offline CDRL demonstrates the best performance among all the algorithms. The reason is that the initial knowledge of the environment does not change up to this time and hence the additional training data generated by offline CDRL is reliable and can help the learning process. Hybrid CDRL outperforms online CDRL before $t = 1600s$ again due to the extra training data generated in the offline part.

Within the duration when Target 2 is in the hazardous region, i.e. between $t = 1600s$ and $t = 2500s$, both online CDRL and hybrid CDRL are able to sense the change in the environment via the online reward and achieve the optimal performance among all the algorithms. We notice that the cost of offline CDRL dramatically increases when the environment changes because offline CDRL is trained in the virtual environment and does not sense the actual environment via online reward.

When there is no target in the hazardous region, we observe similar performance levels achieved by equal allocation and random allocation. Equal allocation and random allocation can also achieve similar performance compared to that of the learning algorithms after around $t = 2800s$. One reason for this is that the two targets have similar distances to the radar and hence equal allocation is desired during this time.

D. Test Case III

In the previous two test cases, all the targets are assumed to have the same features, i.e., they have the same variances for the measurement noise and the maneuverability noise. This assumption is made for validating the effectiveness of the proposed framework. In particular, our aim has been to analyze whether the CDRL agent can adapt to varying target distances and environment properties, and learn effective resource allocation policies. However, in practice, the targets can have different maneuvers, and different levels of measurement noise can be experienced for different targets due to e.g., weather conditions, different environmental settings.

TABLE II
TARGETS' PROPERTIES FOR CASE III

	Target 1	Target 2
$\sigma^2_{r,0}$ (m^2)	2500	9
$\sigma^2_{\theta,0}$ (rad^2)	1e-2	4e-4
σ_w $((m/s^2)^2)$	2500	9

In order to address these scenarios, we consider Case III, which is similar to Case II but the targets have different maneuverability and measurement noise variances. The properties of the two targets are listed in Table II. In this section, we compare the proposed algorithm with two additional benchmarks:

- Distance-based time allocation: Under the budget constraint, the dwell time allocated to Target i is proportional to its estimated distance to the radar.
- Policy rollout: An offline time allocation algorithm proposed in [7], which assumes that the properties of the targets are known beforehand.

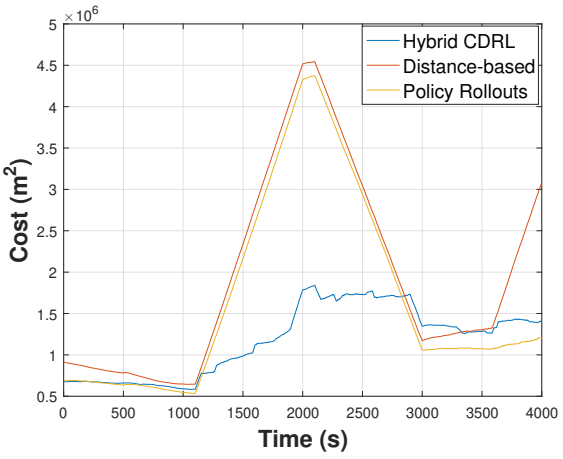

Fig. 17. Case III: Cost Comparison

Comparison of the performance of the proposed hybrid CDRL with those of distance-based and policy rollout resource allocation strategies is provided in Fig. 17, which plots the

total tracking costs. Same as in Case II, Target 2 is within the hazardous region between $t = 1600s$ and $t = 2500s$. Comparing the proposed hybrid CDRL with the distance-based approach, it can be seen that the distance-based approach is not preferred in Case III because a target can have a low tracking cost even when it is far from the radar (due to having lower noise variances). Note that this is different from the previous two cases where the targets have the same noise variances and the distance is the dominant factor in the cost function.

Policy rollout is an offline algorithm that has full prior knowledge of the environment and it performs an exhaustive search considering every feasible action and computes the corresponding cost based on the prior knowledge. Then, the action with the lowest cost is selected. According to Fig. 17, policy rollout achieves the best performance when the environment stays the same, matching its prior knowledge. However, when there is a target moving in the hazardous region (from $t = 1600s$ to $t = 2500s$) and an unexpected penalty is given, the policy rollout algorithm is not able to sense this sudden change in the environment and achieves a degraded performance compared to the proposed hybrid CDRL algorithm.

E. Hybrid CDRL with a Single DQN

Heretofore, the proposed hybrid CDRL algorithm has been implemented in a multi-agent manner, i.e., there are N DQNs, each of which is assigned to tracking one of the N targets. However, the proposed approach can also be implemented with a single DQN to provide scalability in the algorithm and lower the computational complexity. In this subsection, we demonstrate this by implementing hybrid CDRL with a single DQN and evaluating its performance in Test Case II.

Specifically, due to the fact that the state and the action spaces for different targets are the same, all N agents can share a single DQN. At time t, different from the implementation with multiple DQNs, each agent determines its action with the shared DQN. The experience $\{\mathbf{s_t^n}, a_t, r_t, \mathbf{s_{t+1}^n}\}_{n=1}^{N}$ is stored into the experience replay buffer of the shared DQN, and we perform N updates for the shared DQN. In the experiments with a single DQN, we choose the mini-batch size as 32.

Fig. 18 compares different implementations of hybrid CDRL in Test Case II in terms of the tracking cost. We first notice that hybrid CDRL with multiple DQNs achieves the lowest cost and hence the best performance. Expectedly, training and utilizing a separate DQN for each target provides an improved performance. Using a single DQN to decide on the resource allocation for tracking multiple targets leads to a slight increase in the cost but this can be preferred if reducing the computational complexity is critical. In the implementation with a single DQN, we also vary the size of the replay buffer to identify its impact. We observe that as the buffer size increases, the performance degrades when both targets are outside of the hazardous region. This can be attributed to the fact that larger replay buffer would include outdated experience in a highly dynamic environment, negatively influencing the decision making at the DQN. When a target is in the hazardous

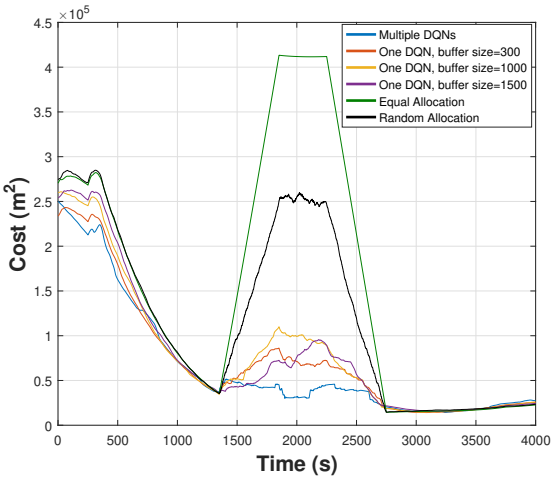

Fig. 18. Cost Comparison Between Multiple DQNs and Single DQN

region, we again note (similarly as in Fig. 16) that equal and random allocation strategies perform especially poorly. In this more challenging environment, performance with a single DQN is significantly better than equal and random allocation schemes while still achieving a cost higher than that of the multi-DQN model. We also observe that having the larger buffer size of 1500 initially leads to smaller cost between $t = 1500$ and $t = 2000$ compared to having the buffer size of 300 and 1000.

Overall, we conclude that implementing the proposed algorithm with a single DQN is feasible but it can lead to an increase in the tracking cost, while decreasing the computational complexity compared to the implementation with N DQNs. The tradeoff between tracking accuracy and computational complexity can be addressed by choosing between the implementation with multiple DQNs and that with a single DQN.

VII. Conclusions

In this work, we have proposed a novel hybrid constrained deep reinforcement learning (hybrid CDRL) framework to address resource allocation in multi-target radar tracking under budget constraints. In particular, we have utilized EKF for target tracking and addressed effective radar time management via hybrid CDRL with the goal to minimize the total tracking cost (and hence maximize the total target tracking accuracy). The proposed hybrid CDRL includes both online and offline learning. Offline CDRL generates training data in a virtual environment to support the algorithm in a fast-varying environment and online CDRL can sense the changes in the environment via the online reward mechanism. With the developed framework, we have shown that dynamic and efficient radar resource allocation can be achieved via hybrid CDRL. More specifically, simulation results show that both online and offline components are essential in the proposed hybrid CDRL framework. In Test Case I, performance of the online CDRL is not as stable as hybrid CDRL due to the

lack of sufficient training data. In Test Case II, performance of the offline CDRL fails to handle the sudden change in the environment since it does not involve a mechanism to incorporate the feedback from the real environment. The combination of both online and offline CDRL, i.e., hybrid CDRL, demonstrates strong performance in both cases and can achieve a stable performance while adapting to the dynamic environment. We have also compared the implementation of the proposed framework with multiple DQNs and a single DQN. Numerical results show that the proposed algorithm with a single DQN can result in slightly higher tracking costs compared to the setting of multiple DQNs. However, the computational complexity is reduced significantly.

REFERENCES

[1] A. Orman, C. N. Potts, A. Shahani, and A. Moore, "Scheduling for a multifunction phased array radar system," *European Journal of operational research*, vol. 90, no. 1, pp. 13–25, 1996.

[2] J. Butler, A. Moore, and H. Griffiths, "Resource management for a rotating multi-function radar," in *Radar 97 (Conf. Publ. No. 449)*. IET, 1997, pp. 568–572.

[3] A. Deligiannis, A. Panoui, S. Lambotharan, and J. A. Chambers, "Game-theoretic power allocation and the nash equilibrium analysis for a multistatic mimo radar network," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6397–6408, 2017.

[4] A. Charlish, F. Hoffmann, C. Degen, and I. Schlangen, "The development from adaptive to cognitive radar resource management," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 6, pp. 8–19, 2020.

[5] Z. Ji and K. R. Liu, "Cognitive radios for dynamic spectrum access-dynamic spectrum sharing: A game theoretical overview," *IEEE Communications Magazine*, vol. 45, no. 5, pp. 88–94, 2007.

[6] S. Haykin, "Cognitive radar: a way of the future," *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 30–40, 2006.

[7] M. Schöpe, H. Driessen, and A. Yarovoy, "A constrained POMDP formulation and algorithmic solution for radar resource management in multi-target tracking," *ISIF Journal of Advances in Information Fusion*, vol. 16, no. 1, p. 31, 2021.

[8] T. de Boer, M. I. Schöpe, and H. Driessen, "Radar resource management for multi-target tracking using model predictive control," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. IEEE, 2021, pp. 1–8.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] C. E. Thornton, M. A. Kozy, R. M. Buehrer, A. F. Martone, and K. D. Sherbondy, "Deep reinforcement learning control for radar detection and tracking in congested spectral environments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1335–1349, 2020.

[11] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, "Reinforcement learning for adaptable bandwidth tracking radars," *IEEE Transactions on Aerospace Electronic Systems*, vol. 56, no. 5, pp. 3904–3921, 2020.

[12] F. Meng, K. Tian, and C. Wu, "Deep reinforcement learning-based radar network target assignment," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 315–16 327, 2021.

[13] W. B. Haskell and R. Jain, "Dominance-constrained markov decision processes," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 5991–5996.

[14] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1889–1897.

[16] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[17] H. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3703–3712.

[18] F. Fioretto, T. Mak, F. Baldo, M. Lombardi, and P. Van Hentenryck, "A Lagrangian dual framework for deep neural networks with constraints," *arXiv preprint arXiv:2001.09394*, 2020.

[19] M. Eisen, C. Zhang, L. F. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2775–2790, 2019.

[20] S. Khairy, P. Balaprakash, L. X. Cai, and Y. Cheng, "Constrained deep reinforcement learning for energy sustainable multi-UAV based random access IoT networks with NOMA," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 4, pp. 1101–1115, 2020.

[21] W. Koch, "Adaptive parameter control for phased-array tracking," in *Signal and Data Processing of Small Targets 1999*, vol. 3809. SPIE, 1999, pp. 444–455.

[22] H. Meikle, *Modern radar systems*. Artech House, 2008.

[23] G. Welch, G. Bishop *et al.*, "An introduction to the Kalman filter," 1995.

[24] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[25] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009, vol. 48.

Ziyang Lu received the B.S. degree in electrical engineering from the University of Liverpool, U.K., in 2016, and the M.S. degree in electrical engineering from Syracuse University in 2018, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science. His research interests are in the areas of wireless communication and machine learning.

M. Cenk Gursoy received the B.S. degree with high distinction in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 1999 and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 2004. He was a recipient of the Gordon Wu Graduate Fellowship from Princeton University between 1999 and 2003. He is currently a Professor in the Department of Electrical Engineering and Computer Science at Syracuse University. His research interests are in the general areas of wireless communications, information theory, communication networks, signal processing, optimization and machine learning. He is an Editor for IEEE Transactions on Communications, and an Area Editor for IEEE Transactions on Vehicular Technology. He is on the Executive Editorial Committee of IEEE Transactions on Wireless Communications. He also served as an Editor for IEEE Transactions on Green Communications and Networking between 2016–2021, IEEE Transactions on Wireless Communications between 2010–2015 and 2017–2022, IEEE Communications Letters between 2012–2014, IEEE Journal on Selected Areas in Communications - Series on Green Communications and Networking (JSAC-SGCN) between 2015-2016, Physical Communication (Elsevier) between 2010–2017, and IEEE Transactions on Communications between 2013–2018. He has been the co-chair of the 2017 International Conference on Computing, Networking and Communications (ICNC) - Communication QoS and System Modeling Symposium, the co-chair of 2019 IEEE Global Communications Conference (Globecom) - Wireless Communications Symposium, the co-chair of 2019 IEEE Vehicular Technology Conference Fall - Green Communications and Networks Track, and the co-chair of 2021 IEEE Global Communications Conference (Globecom), Signal Processing for Communications Symposium. He received an NSF CAREER Award in 2006. More recently, he received the EURASIP Journal of Wireless Communications and Networking Best Paper Award, 2020 IEEE Region 1 Technological Innovation (Academic) Award, 2019 The 38th AIAA/IEEE Digital Avionics Systems Conference Best of Session (UTM-4) Award, 2017 IEEE PIMRC Best Paper Award, 2017 IEEE Green Communications & Computing Technical Committee Best Journal Paper Award, UNL College Distinguished Teaching Award, and the Maude Hammond Fling Faculty Research Fellowship. He is a Senior Member of IEEE, and is the Aerospace/Communications/Signal Processing Chapter Co-Chair of IEEE Syracuse Section.