Pulkkinen, Petteri; Aittomaki, Tuomas; Strom, Anders; Koivunen, Visa

Time Budget Management in Multifunction Radars Using Reinforcement Learning

Published in:
2021 IEEE Radar Conference

DOI:
10.1109/RadarConf2147009.2021.9455344

Published: 07/05/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Pulkkinen, P., Aittomaki, T., Strom, A., & Koivunen, V. (2021). Time Budget Management in Multifunction Radars Using Reinforcement Learning. In *2021 IEEE Radar Conference: Radar on the Move, RadarConf 2021* Article 9455344 (Proceedings of the IEEE Radar Conference; Vol. 2021-May). IEEE. https://doi.org/10.1109/RadarConf2147009.2021.9455344

Time Budget Management in Multifunction Radars Using Reinforcement Learning

Petteri Pulkkinen*†, Tuomas Aittomäki*, Anders Ström‡ and Visa Koivunen*
*Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland
Email: {petteri.pulkkinen,tuomas.aittomaki,visa.koivunen}@aalto.fi
†Saab Finland Oy, Helsinki, Finland
‡Radar Solutions, Saab AB, Göteborg, Sweden

***Abstract*—An adaptive revisit interval selection (RIS) in multifunction radars is an integral part of efficient time budget management (TBM). In this paper, the RIS problem is formulated as a Markov decision process (MDP) with unknown state transition probabilities and reward distributions. A reward function is proposed to minimize the tracking load (TL) while maintaining the track loss probability (TLP) at a tolerable level. The reinforcement learning (RL) problem is solved using the Q-learning algorithm with an epsilon-greedy policy. Compared to a baseline algorithm, the RL approach was capable of maintaining the tracks while reducing the tracking load significantly.**

***Index Terms*—reinforcement learning, revisit interval selection, adaptive update rate, radar, Q-learning, time budget management**

I. Introduction

Time budget management (TBM) is an important problem in modern multifunction radars that considers allocating and scheduling radar time resources for various radar tasks [1]. Modern multifunction radars typically use phased array technology, such as the active electronically scanned array (AESA), that enables steering the radar beam in any direction with a relatively short delay. Moreover, cognitive radars employ the sense–learn–adapt cycle to exploit past observations and construct situational awareness for continuously improving the radar performance [2]. Thus, intelligent radar TBM algorithms are needed to fully utilize the capabilities of cognitive multifunction radars.

An integral part of the TBM is the selection of the revisit intervals (RIs) for the target tracking tasks [3]–[11]. The RI is the time interval between track update attempts. The radar time resources can be conserved in tracking tasks by using long RIs. However, the RIs should be selected such that the active tracks can be maintained. Short RIs should be used for maneuvering targets to maintain the quality of state estimates at a tolerable level. On the other hand, long RIs should be used for non-maneuvering targets with stable trajectories because the movement is predictable.

Revisit interval selection (RIS) algorithms are typically either residual-based [6], [7] or prediction error covariance matrix (PECM) based algorithms [3], [4], [8]. The residual-based RIS algorithms select the RIs adaptively based on the innovation sequence of a tracking filter. The PECM-based algorithms impose a constraint for the PECM such that the RIS problem can be formulated as an optimization problem. In addition, many other RIS algorithms are based on similar principles of the algorithms mentioned above [5], [9]–[11]. However, the previously proposed RIS algorithms use target, radar, and environment models, making them vulnerable to modeling errors. Consequently, the RIS problem could be approached with reinforcement learning (RL) methods that allow learning and autonomous performance improvements in real-time while avoiding the modeling errors [12].

This paper proposes an RL algorithm to solve the RIS problem. In other words, the RIS problem is formulated as an RL problem and solved using the Q-learning algorithm with an ϵ-greedy policy. The proposed RL algorithm learns to select RIs to minimize the tracking load (TL) [3] while keeping the track loss probability (TLP) [13] at a tolerable level. The RL approach is trained in simulations using benchmark trajectories introduced in [13], and the performance of the proposed algorithm is compared against a baseline algorithm. The proposed method is able to maintain the tracks while significantly reducing the tracking load compared to the baseline solution.

II. Problem description

The multifunction radar of the model assumed in this paper searches for new targets using the search function. After a new target is found, a track is initiated and the tracking loop, shown in Fig. 1, will start. A multiple model estimator, such as the interacting multiple model (IMM) estimator [14, pp. 453–457], is used to predict and filter the target states. The tracking loop would be the same for each target in a multi-target tracking (MTT) scenario. However, a single-target tracking (STT) scenario is assumed in this study such that only one tracking loop occurs at a time. The STT scenario simplifies the RIS problem because the track association does not have to be considered. However, this allows better understanding of the learning performance.

In the tracking loop, an RIS algorithm controls the RI according to the information given for the algorithm. If a track is updated at time instance $t(k)$, then the next update time instance is $t(k+1) = t(k) + t_r(k)$, where $t_r(k)$ is the RI chosen at time instance $t(k)$. The utilization of time resources for a given RI can be quantified with a TL [3] which is defined as follows

$$L_k(t_r) = \frac{\mathbb{E}\left[n_k | t_r\right] \tau_d}{t_r}, \tag{1}$$

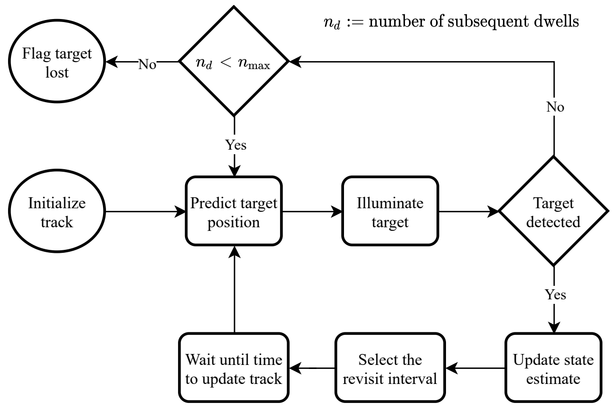


Fig. 1. Tracking loop of the multifunction radar model.

where n_k is the number of dwells needed to obtain a successful detection, and τ_d is the dwell time. To save time resources, the RI is selected to minimize the TL while keeping the TLP at a tolerable level. The TLP [13] is defined as follows

$$P_{\text{loss}}(T_i) = \Pr\{\text{track loss}|T_i, \pi\} \tag{2}$$

which is the probability of losing a specific track T_i given the RIS policy π. In [13], a performance criterion was defined that bounds $P_{\text{loss}}(T_i)$ below a specific value for any T_i. However, here the RIS algorithms are evaluated in terms of the TLP.

III. Reinforcement learning

Reinforcement learning is a method for learning to act in stochastic sequential decision-making problems without an assumed model that describes the problem dynamics [12]. An RL agent interacts with an environment by taking different actions. After taking an action, the agent observes the environment state and receives a scalar reward that quantifies the quality of the action. Given the state of the environment, the agent takes the next action. The learning is based on a trial and error approach, which is controlled through the rewards. The agent independently improves its performance by probing different actions for different states to learn their consequences towards maximizing future rewards.

A reinforcement learning problem is modeled as a Markov decision process (MDP) in which the state transition probabilities and rewards are unknown [12]. The agent needs to find a policy $\pi(a|s)$, which gives the probability for taking action $a \in \mathcal{A}$ given the state $s \in \mathcal{S}$. The optimal policy π^* maximizes the discounted sum of future rewards G_k given any initial state s as follows [12]

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[G_k|S_k = s, \pi\right], \tag{3}$$

where S_k is the state at time instance k and

$$G_k = \sum_{i=0}^{\infty} \lambda^i R_{k+i+1}. \tag{4}$$

The parameter $\lambda \in [0, 1)$ is called discount factor and R_{k+i+1} is the immediate reward received after taking an action at time instance $k + i$. For $\lambda \ll 1$, only near future and immediate rewards have a significant effect on G_k. Otherwise, also the long-term future rewards are weighted with large values. A policy that considers only the immediate rewards is called a myopic policy. In general, myopic policies are sub-optimal for $\lambda > 0$.

Most RL algorithms are based on estimating either state-values or action-values [12]. The state-value function for policy π is defined as follows

$$v_\pi(s) = \mathbb{E}\left[G_k|S_k = s, \pi\right]. \tag{5}$$

Similarly, the action-value function can be written as follows

$$q_\pi(s, a) = \mathbb{E}\left[G_k|S_k = s, A_k = a, \pi\right], \tag{6}$$

where A_k is the action chosen at time instance k. The state- or action-values of an optimal policy are larger or equal compared to the values of any other policy.

A. *Exploration–exploitation trade-off*

An RL agent taking an action to gain more knowledge about the dynamics of the MDP is said to be exploring. Otherwise, the agent is exploiting, which means that the agent chooses the action currently believed to be the best one. The dilemma of finding the balance between exploration and exploitation is called the exploration–exploitation trade-off.

A widely used policy for balancing the exploration–exploitation trade-off is the ϵ-greedy policy [12]. With the ϵ-greedy policy, the agent chooses a random action with probability ϵ. When not choosing the action randomly, the action with the highest estimate of the action-value given the state $s \in \mathcal{S}$ is selected. Therefore, the policy can be written as follows

$$A_k = \begin{cases} \arg\max_{a\in\mathcal{A}} q(s, a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon, \end{cases} \tag{7}$$

where $q(s, a)$ is an estimate of the action-value $q_\pi(s, a)$. The estimate $q(s, a)$ is also known as Q-value. The ϵ-greedy policy cannot achieve optimality with a fixed exploration probability ϵ because random actions will always be taken. One solution is to let ϵ decay gradually.

B. *Q-learning algorithm*

The Q-learning algorithm is a specific off-policy temporal difference (TD) learning algorithm [12]. It is called off-policy algorithm because Q-learning uses different policies to update Q-values (target policy) and take actions while learning (behavior policy). In the Q-learning algorithm, the target policy selects the action with the highest Q-value, and the behavior policy implements the exploration and exploitation trade-off. For example, the behavior policy can be the ϵ-greedy policy.

The Q-learning algorithm updates the Q-values using the following TD update rule

$$q(s, a) \leftarrow q(s, a) + \alpha\left(r + \text{argmax}_{a'} q(s', a') - q(s, a)\right) \tag{8}$$

where $\leftarrow$ is the assignment operator, parameter $\alpha \in (0, 1]$ is called the learning rate, s' is the next state, and r is the

```
1: Initialize $q(s,a),\ \#(s,a) = 0\,\forall\, s \in \mathcal{S}, a \in \mathcal{A}$
2: while learning do
3:     $a \leftarrow \begin{cases} \text{argmax}_{a'} q(s,a') & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$
4:     $r, s' \leftarrow$ take action $a$
5:     $\#(s,a) \leftarrow \#(s,a) + 1$    ▷ # of times $a$ and $s$ visited.
6:     $\alpha \leftarrow \#(s,a)^{-\omega}$
7:     $q(s,a) \leftarrow q(s,a) + \alpha(r + \lambda \max_{a'} q(s',a') - q(s,a))$
8:     $s \leftarrow s'$
9: end while
```

Fig. 2. The Q-learning algorithm with an ϵ-greedy policy [12] and asynchronously decayed learning rate [15].

immediate reward received after taking action a. The target policy is proven to converge to the optimal policy if the behavior policy explores actions with a probability larger than zero and α is sufficiently small [12]. The learning rate α can be made decaying by using an asynchronous strategy with a parameter $\omega \in [0.5, 1]$ as proposed in [15]. The Q-learning algorithm with the ϵ-greedy policy and the asynchronously decaying α is shown in Figure 2.

IV. RL formulation of the RIS problem

A. Actions

The RL agent selects the RI from a set of intervals. In other words, the selected RI is $t_r(k) = A_k$, where $A_k \in \{a_{\text{min}}, ..., a_{\text{max}}\}$ is the action taken at time instance k. The action space is discretized such that the resulting RIs are equally spaced. In addition, the minimum RI $t_{\text{min}} = a_{\text{min}}$ and the maximum RI $t_{\text{max}} = a_{\text{max}}$ are defined. Therefore, the action space can be written as follows

$$\mathcal{A} = \{\frac{nt_{\text{max}} + (N_{\text{a}} - n - 1)t_{\text{min}}}{N_{\text{a}} - 1}\}_{n=0}^{N_{\text{a}}-1}, \tag{9}$$

where N_{a} is the size of the action space. Smaller N_{a} enables faster learning since there are fewer actions to be explored. However, larger N_{a} may enable higher performance since a larger variety of RIs can be used.

B. Rewards

The agent's overall objective is to minimize the time budget needed to maintain a track and prevent track losses based on the cost associated to losing a track. The objective should be reflected in the immediate or cumulative rewards given for the RL agent, as shown in (4). The proposed immediate rewards are based on the TL in (1) and the track loss cost c_{loss} as follows

$$r = \begin{cases} -c_{\text{loss}} & \text{if target lost} \\ -\frac{n_k}{t_r}\tau_d & \text{otherwise.} \end{cases} \tag{10}$$

In the reward function, $\frac{n_k}{t_r}\tau_d$ is the TL in (1) without expectation on the number of dwells. The discount factor λ in (4) can be used to control how much the future TL and the probability of losing a track are considered.

The value for c_{loss} can be selected using a heuristic design rule based on a simplified MDP with two states. The states of the MDP represent if a target is lost or not. An agent needs to select an action from two competing actions. One of the actions, denoted as a_1, has a TL of L_{ref}, and when taking it, the track cannot be lost. On the other hand, the track is lost with probability 1 if the other action a_2 is taken. Assuming that tracking a target with TL of L_{ref} is equally bad as losing the track, the parameter c_{loss} should be defined such that

$$c_{\text{loss}} = \sum_{i=0}^{\infty} \lambda^i L_{\text{ref}}. \tag{11}$$

Equation (11) can be rewritten into form

$$c_{\text{loss}} = \frac{1}{1-\lambda} L_{\text{ref}} \tag{12}$$

by using the geometric series.

C. States and observations

The RIS problem is a partially observable Markov decision process (POMDP) meaning that the observations are not Markovian [16]. Any POMDP can be converted to a continuous state MDP by using belief states as states. However, the tabular Q-learning algorithm [12] requires the states to be discrete making the state-space cardinality grow exponentially as the number of state variables increase. RL algorithms can, however, tolerate approximately Markovian states. Thus, only relevant information needs to be extracted to prevent the exponentially growing cardinality. Later on, those observations will be referred to as states or agent states, interchangeably.

We propose various observations that can be used as agent states and the performance of those will be evaluated in Section VI-A. An important source of information from the tracking filters is the innovation sequence. The simplest way to utilize the innovation sequence in discrete RL algorithms is to use the most recent innovation as a state. The size of the state space can be further reduced by using only the innovation in the target angle estimates, assuming that the predicted target range is not critical for steering the radar beam in the correct direction. Another option is to use the mode probabilities of an IMM estimator as the states. The number of states is manageable as long as the number of the quantized mode probabilities and the target motion models remain sufficiently small.

The state spaces based on the innovation or the mode probabilities can be further extended by augmenting the most recent RI to the state presentation. Also, two additional states are included in the state space. One is for a lost target and the other for a track initiation. The latter state enables the agent to learn to select RI at the beginning of the tracking task because the tracking filter might have unreliable estimates.

Each continuous state variable s_m is discretized into $N_q^{(m)}$ states by linearly quantizing observations between defined minimum l_m and maximum u_m and reserving two states for observations below l_m and above u_m. Thus, the number of states is $N_s = 2 + \prod_{m=0}^{M-1} N_q^{(m)}$ if the previous RI is not included in the state. If the previous RI is included into the state then $N_s = 2 + N_a \prod_{m=0}^{M-1} N_q^{(m)}$.

V. Numerical examples

The developed RL approach for the RIS problem is evaluated in Monte Carlo (MC) simulations. The simulation scenario contains a monostatic radar in two-dimensional (2-D) space at the position $x = 0$ and $y = 0$. The probability of detection is simulated for Swerling I target as follows [3]

$$P_{\text{d}} = P_{\text{fa}}^{\frac{1}{1+SNR}}, \tag{13}$$

where $P_{\text{fa}} = 1\text{e}{-6}$ is the imposed probability of false alarm constraint. However, P_{fa} is only utilized to calculate the probability of detection and false alarms are not included in the simulations. The target is considered lost after $n_{\text{max}} = 20$ subsequent dwells without a detection.

The angle error in illuminating the target is assumed to affect the signal-to-noise ratio (SNR) based on the following equation [3]

$$SNR = SN_0 \exp\left(-\ln 2 \frac{(\hat{\theta} - \theta)^2}{B^2}\right), \tag{14}$$

where $SN_0 = 50$ is the boresight SNR in a linear scale, $B = 0.02$ is half of the -3dB beamwidth in radians, θ is target angle and $\hat{\theta}$ is the boresight angle. In addition, the boresight SNR SN_0 and half of the -3dB beamwidth B are constant in the simulations.

The target trajectories are based on a benchmark dataset introduced in [13]. However, the spatial dimension of the trajectories is reduced from three-dimensional (3-D) to 2-D space. The six different target trajectories include different kinds of motion, such as constant-g turns and straight motion with a constant or non-constant velocity.

A. *CVCA-IMM estimator*

The tracking filter employed in the simulations is an IMM estimator that uses two linear Kalman filters configured for two target motion models. The used target motion models are the constant velocity (CV) and constant acceleration (CA) models described in [14, pp. 273–275]. Only position measurements in Cartesian coordinates are considered in both filters, and the standard conversion [14, pp. 397-399] from the polar coordinates to the Cartesian coordinates is used. The standard deviations of the azimuth angle error σ_θ and the range error σ_r are 0.002 and 2.85, respectively. The mode transition probability, denoted as μ, from CV to CA and CA to CV is set equal. This IMM estimator configuration will be called the CVCA-IMM estimator.

The CVCA-IMM estimator has three parameters to be tuned. These parameters are the acceleration noise variance for the CV model σ^2_{CV}, the acceleration noise variance for the CA model σ^2_{CA}, and the mode transition probability μ. The mode transition probability is defined for a processing time step $\Delta t = 0.1$, which the tracker uses to predict the future states recursively. The parameters $\mu = 0.028$, $\sigma^2_{\text{CV}} = 747.5$ and $\sigma^2_{\text{CA}} = 74.7$ were found empirically by simulating the CVCA-IMM estimator with different parameter combinations.

TABLE I
Proposed RL agent configurations.

Agent ID	**State space**	$\boldsymbol{\lambda}$	$\boldsymbol{\omega}$
0	innovation	0.0	1.0
1	innovation	0.7	0.75
2	innovation & previous RI	0.0	1.0
3	innovation & previous RI	0.7	0.75
4	mode probability	0.0	1.0
5	mode probability	0.7	0.75
6	mode probability & previous RI	0.0	1.0
7	mode probability & previous RI	0.7	0.75

B. *Baseline algorithm used in comparisons*

A baseline algorithm is utilized to compare the RL approach against a conventional RIS algorithm. The baseline RIS algorithm is based on an approach proposed in [4]. This algorithm is a PECM-based algorithm that optimizes

$$\begin{aligned} &\max\nolimits_{t_r} t_r \\ \text{s.t.}\quad &\sigma_\theta(t + t_r|t) \le V_0 B, \end{aligned} \tag{15}$$

where V_0 is called track sharpness parameter, and $\sigma_\theta(t+t_r|t)$ is the standard deviation of the error of the predicted azimuth angle. The longest RI to satisfy the constraint is found by propagating the predictive equations of the CVCA-IMM estimator with the processing time step Δt. The track sharpness V_0 affects the TLP and needs to be tuned for the specific IMM estimator. The performance with different values of V_0 is discussed in Section VI-B.

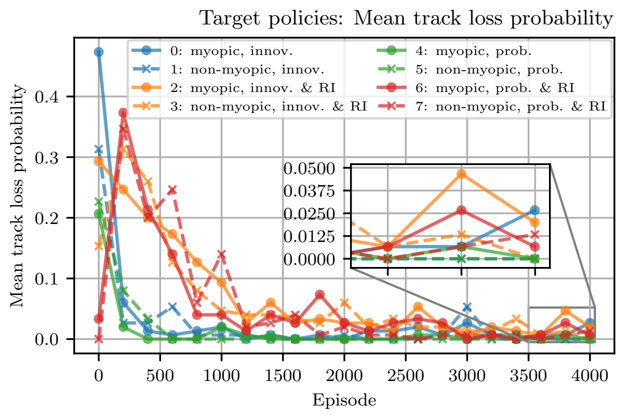
C. *Reinforcement learning agent configurations*

Various RL agents are configured such that it is possible to identify effective state spaces from the states discussed in Section IV-C and to determine if using a myopic policy is sufficient. All the agents use the Q-learning algorithm presented in Figure 2.

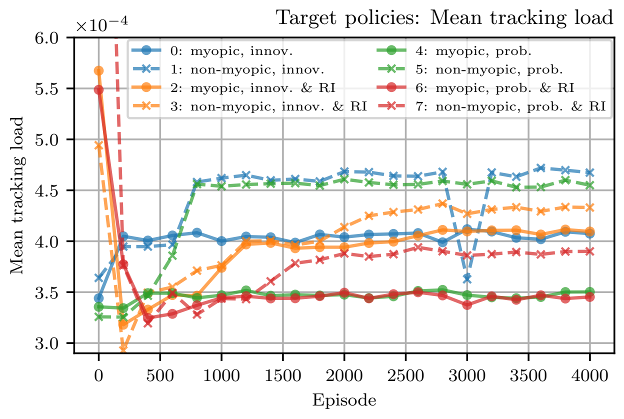
The size of the action space $N_{\text{a}} = 10$, the minimum RI $t_{\text{min}} = 0.1s$, and the maximum RI $t_{\text{max}} = 7.5s$. State variables used for the agents are the most recent innovation, the probability of the CV mode, and the previous RI. Combinations of these state variables will be used as states, as shown in Table I. The continuous state variables are quantized into $N_q = 10$ discrete bins. Therefore, the number of states N_s is 12 for the state spaces not the augmenting with the RI, and $N_s = 102$ otherwise. For the innovation, the minimum and maximum quantization values are $l = 0.05B$ and $u = 0.5B$. For myopic agents the discount factor $\lambda = 0$ and the learning rate decay parameter $\omega = 1.0$, and for non-myopic agents $\lambda = 0.7$ and $\omega = 0.75$ are used.

VI. Simulation results

The RIS algorithms are evaluated with TL and TLP based measures. Especially, the mean tracking load (MTL) and the mean track loss probability (MTLP) are defined to summarize the algorithm performance. The MTL is the expected arithmetic mean of TLs when a track is sampled from a probability



(a) Mean track loss probability (MTLP).



(b) Mean tracking load (MTL).

Fig. 3. Target policy performance with different numbers of learning episodes. The agent ID=4 achieves a great performance in terms of the learning speed, the MTL and the MTLP.

distribution $P_T(i)$ and i is an index of a track. Similarly, the MTLP is the TLP marginalized with the distribution $P_T(i)$. Additionally, Euclidean distance between target position and predicted target position is referred to as position prediction error (PPE).

A. *Reinforcement learning agent training*

The RL agents shown in Table I were trained by uniformly sampling the target trajectories from the employed benchmark trajectories. Therefore, the sampling distribution $P_T(i)$ is a discrete uniform distribution. Each agent configuration was trained once for 4000 episodes. An episode means tracking a target for the duration of a benchmark trajectory that ends immediately if the target is lost. The parameter L_{ref} was 0.2 such that, according to (12), the track loss cost $c_{\text{loss}} = 0.2$ for myopic agents and $c_{\text{loss}} \approx 0.667$ for non-myopic agents. The exploration probability was initially $\epsilon = 0.2$, and it was decayed between each episode. The decay factor was 0.99868, which resulted in $\epsilon = 0.001$ in the last training episode.

Figure 3a shows the MTLPs obtained by the target policies with constant Q-values at different stages of the training. These values were computed over 150 MC iterations per each stage. A common decreasing trend can be seen in Fig. 3a that demonstrates the improvement achieved via learning. In addition, two distinct clusters can be seen in episodes approximately up to 1500. The agents appear in the clusters based on whether or not the state space has been augmented with the previous RI. Moreover, the agents with larger state spaces seem to learn slower. Figure 3a also shows that most myopic agents have larger MTLPs than the non-myopic counterparts (with an equal state space).

Figure 3b shows the MTLs obtained by the target policies. A small number of training episodes were required to achieve low MTL, but it can also be seen that MTL increase as the agents learn more. The myopic agents have significantly lower MTL than non-myopic counterparts. Also, the non-myopic agents with the augmented state spaces achieve significantly smaller MTLs than the corresponding non-myopic agents not augmenting the previous RI. However, no significant effect on MTL can be seen for myopic agents when augmenting state presentations with the previous RI. Assessing whether augmenting state spaces with the previous RI can improve the performance significantly is difficult since MTLP in Fig. 3a increases similarly as the MTL decreases.

It is challenging to learn to maintain the tracks because the track loss penalties are sparse. Therefore, having more state–action pairs to be explored causes the clusters in Fig. 3a. In contrast, the tracking load penalties are frequent, and hence learning to choose short RIs to minimize the MTL requires a small number of training episodes. The slight growth of MTL shown in Fig. 3b is caused by the fact that the agents become more conscious of losing tracks. In addition, the non-myopic agents lose tracks less often than the myopic agents because the long-term consequences for the actions are considered. Consequently, the myopic agents have higher MTLP and lower MTL when compared to non-myopic counterparts, as shown in Figures 3a and 3b.

The behavior policy performance is somewhat similar compared to a target policy during the first training episodes. However, the random actions increases the MTL significantly because those actions are taken with high probability before the exploration probability ϵ has considerably decayed. The effect to MTLP is not significant since the converge for target policies is already slow.

B. *Performance against the baseline algorithm*

The desired attributes of an RL agent are a fast learning speed, a low MTLP, and a low MTL. Based on the results in Section VI-A, the agent ID $= 4$ meets these requirements well. Therefore, this RL agent configuration was trained with various values of c_{loss} to compare the performance against the baseline algorithm. Figure 4 shows the MTL as a function of MTLP for the RL agent (ID=4) and the baseline algorithm with different values of V_0. It can be seen that the RL approach achieves a lower MTL than the baseline algorithm when MTLP is below 0.02. However, for MTLPs larger than approximately 0.02, the RL agent's MTL is slightly higher compared to the baseline algorithm. In other words, the RL

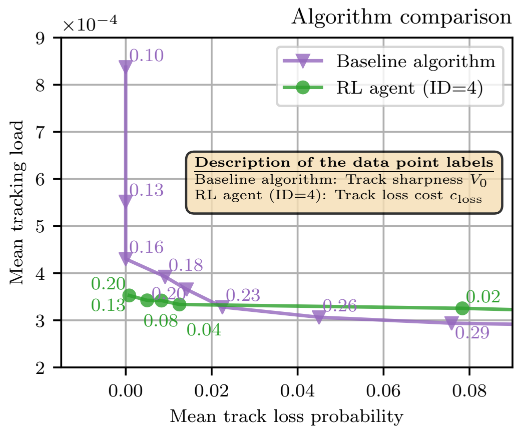


Fig. 4. The MTL as a function of MTLP for the baseline algorithm and the RL agent (ID=4). The RL agent achieves lower MTLs than the baseline algorithm with MTLPs below 0.02.

agent is more reliable in maintaining the tracks with a low MTL. It can also be seen that the MTL of the baseline algorithm is quite sensitive to the selection of V_0 parameter, especially when selecting small values of V_0 in order to achieve a low MTLP.

The algorithms using parameters $c_{\text{loss}} = 0.2$ and $V_0 = 0.164$ were selected for a closer examination. With the corresponding parameters, the RL agent lost the track 4 once in 200 MC iterations. The other tracks were not lost a single time. The baseline algorithm did not lose any tracks in the MC simulations.

Table II shows the RL agent performance relative to the baseline algorithm performance. The performance is compared using the mean and peak values of TLs and position prediction errors (PPEs) for each track $T_i \in \{T_1, \dots, T_6\}$. The RL agent achieves lower mean and peak TLs when compared to the baseline algorithm. Notably, the TL peaks are reduced on average by a factor of 0.55. The mean PPE is not increased significantly in most of the tracks, but the PPE peaks are slightly higher. An exception is the track 4, in which the RL agent was struggling the most to maintain the track. Therefore, the mean PPE was higher by a factor of 1.43 and the peak PPE by a factor of 2.25 when compared to the baseline algorithm. In general, tracks at a further distance are maintained more reliably than the tracks closer to the radar.

VII. Conclusions

A reinforcement learning (RL) method based on the Q-learning algorithm was proposed to address the revisit interval selection (RIS) problem of multifunction radars. The method utilizes its earlier tracking experience to learn a RIS policy in real-time to minimize the tracking load (TL) while keeping the risk of losing tracks at a tolerable level.

Differently configured RL agents were proposed and evaluated using Monte Carlo simulations. The results indicate that the size of the state space affects the learning speed significantly. Also, non-myopic agents were generally better

TABLE II
RL agent performance relative to the baseline algorithm

		T_1	T_2	T_3	T_4	T_5	T_6	Avg
TL	**Mean**	0.98	0.77	0.89	0.60	0.95	0.81	0.83
	Peak	0.64	0.47	0.58	0.51	0.57	0.50	0.55
PPE	**Mean**	1.01	1.13	1.05	1.43	1.01	1.23	1.14
	Peak	1.44	1.31	1.27	2.25	0.67	1.19	1.36

in maintaining the tracks. However, the myopic agent using the mode probabilities as states achieved the most promising performance in terms of mean track loss probability (MTLP), mean tracking load (MTL) and ability to learn fast.

The most promising RL agent was also compared against a conventional RIS algorithm. The MTL of the RL agent was lower compared to the baseline algorithm when evaluated with low MTLP values. Thus the proposed RL algorithm maintained the tracks with a lower TL. It was especially shown that the proposed RL algorithm had smaller TL peaks compared to the baseline algorithm.

References

[1] P. Moo and Z. Ding, *Adaptive Radar Resource Management*, 1st ed. London, UK: Academic Press, 2015.

[2] S. Haykin, "Cognitive radar: a way of the future," *IEEE Signal Process. Mag.*, vol. 23, no. 1, pp. 30–40, Jan. 2006.

[3] G. van Keuk and S. S. Blackman, "On phased-array radar tracking and parameter control," *IEEE Trans. on Aerosp. and Electron. Syst.*, vol. 29, no. 1, pp. 186–194, Jan. 1993.

[4] E. Daeipour, Y. Bar-Shalom, and X. Li, "Adaptive beam pointing control of a phased array radar using an IMM estimator," in *Proc. of 1994 Amer. Control Conf. - ACC '94*, vol. 2, Jun. 1994, pp. 2093–2097.

[5] H. Benoudnine, M. Keche, A. Ouamri, and M. S. Woolfson, "Fast adaptive update rate for phased array radar using IMM target tracking algorithm," in *2006 IEEE Int. Symp. on Signal Process. and Inf. Technol.*, Aug. 2006, pp. 277–282.

[6] Cheng Ting, He Zi-shu, and Tang Ting, "An IMM-based adaptive-update-rate target tracking algorithm for phased-array radar," in *2007 Int. Symp. on Intell. Signal Process. and Communication Syst.*, Nov. 2007, pp. 854–857.

[7] S. H. Baek, H. Seok, K. H. Park, and J. Chun, "An adaptive update-rate control of a phased array radar for efficient usage of tracking tasks," in *2010 IEEE Radar Conf.*, May 2010, pp. 1214–1219.

[8] A. Charlish and F. Hoffmann, "Anticipation in cognitive radar using stochastic control," in *2015 IEEE Radar Conf.*, May 2015, pp. 1692–1697.

[9] F. Masoumi-Ganjgah, R. Fatemi-Mofrad, and N. Ghadimi, "Target tracking with fast adaptive revisit time based on steady state IMM filter," *Digit. Signal Process.*, vol. 69, pp. 154–161, 2017.

[10] M. Pilté, S. Bonnabel, and F. Barbaresco, "Fully adaptive update rate for non-linear trackers," *IET Radar, Sonar Navigation*, vol. 12, no. 12, pp. 1419–1428, 2018.

[11] J. M. Christiansen, K. E. Olsen, and G. E. Smith, "Fully adaptive radar for track update-interval control," in *2018 IEEE Radar Conf.*, Apr. 2018, pp. 0400–0404.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[13] W. D. Blair, G. A. Watson, T. Kirubarajan, and Y. Bar-Shalom, "Benchmark for radar allocation and tracking in ECM," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 4, pp. 1097–1114, Oct. 1998.

[14] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, 1st ed. New York: Wiley, 2001.

[15] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *J. of Mach. Learning Research*, vol. 5, pp. 1–25, 2003.

[16] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*, 1st ed. Cambridge, UK: Cambridge University Press, 2016.