REINFORCEMENT LEARNING-BASED DYNAMIC SCHEDULING FOR THREAT EVALUATION

Nimrod Lilith and Kutluyıl Doğançay

School of Electrical and Information Engineering
University of South Australia
Mawson Lakes SA 5095
phone: + (61) 8 83023320, fax: + (61) 8 83023384
email: Nimrod.Lilith@unisa.edu.au, k.dogancay@ieee.org

ABSTRACT

A novel reinforcement learning-based sensor scan optimisation scheme is presented for the purpose of multi-target tracking and threat evaluation from helicopter platforms. Reinforcement learning is an unsupervised learning technique that has been shown to be effective in highly dynamic and noisy environments. The problem is made suitable for the use of reinforcement learning by its casting into a "sensor scheduling" framework. An innovative action exploration policy utilising a Gibbs distribution is shown to improve agent performance over a more conventional random action selection policy. The efficiency of the proposed architecture in terms of the prioritisation of targets is illustrated via simulation examples.

1. INTRODUCTION AND BACKGROUND

Reinforcement learning, also known as neuro-dynamic programming [1], is an optimisation technique that learns the best actions to take by interacting with the environment [2]. This paper investigates the application of reinforcement learning to a particular sensor scanning problem in helicopter missions. Here the sensor is an electro-optical vision device on-board a helicopter. The sensor aims to detect ground targets of different types (such as humans, trucks, armoured vehicles) in a region of interest by means of scanning that permits only one small region to be visible at a given time. Based on mathematical modelling of target movements and target priorities we examine best scanning strategies for the sensor compared with deterministic round-robin and random scanning patterns.

Many practical problems involving a selection out of several available options in order to maximise a reward function can be defined as a multi-armed bandit problem [3]. Dynamic sensor allocation is one example. Bandwidth limitations for communicating with several sensors often impose a limit on how many sensors can be accessed at a given time. Ideally all sensors must be utilised in order to make the best decision. However, when only a small subset of sensors can be accessed at a given time, a decision needs to be made to select the "best" sensors to be employed at the next time instant based on current and previous sensor selections and the reward associated with them. Thus, the main objective of dynamic sensor allocation is to maximise a reward function that can be obtained from a limited number of sensors available by finding an optimal sequence of sensor selection at discrete time instants. The application of the multi-armed bandit approach to sensor scheduling has been reported in [4]. A weakness of multi-armed bandits is that targets not selected are assumed to have frozen states. This restrictive assumption is not necessary for reinforcement learning.

The dynamic sensor allocation problem has numerous applications in a wide-range of diverse fields such as electronic warfare and telecommunications. Reinforcement learning provides an on-line computationally inexpensive solution to dynamic programming problems that suffer from the curse of dimensionality due to state-space explosion. What is more, reinforcement learning, unlike dynamic programming, assumes no *a priori* knowledge of the environment where it operates. The existing algorithms for dynamic sensor allocation suffer from the curse of dimensionality [2] and make restrictive assumptions such as Markovian state transitions.

In this paper we present a novel reinforcement learning-based dynamic sensor scan optimisation scheme that prioritises individual targets based upon threat evaluation. The adopted approach exploits a novel hybrid exploration scheme utilising a Gibbs distribution in an ε-greedy algorithm. The new algorithm is shown to be capable of learning in a fully on-line manner without any training. By way of computer simulations we demonstrate the effectiveness of the proposed learning architecture.

2. SENSOR SCAN OPTIMISATION

Army helicopters utilise electro-optic sensors to detect and identify different targets. Two commonly used electro-optic sensors are forward looking IR (FLIR) with a range of 1–6 km range depending on the terrain and the human eye. The targets of interest are

- humans (smallest);
- light vehicles (small);
- armoured vehicles (large);
- ADUs, i.e., air defence units (largest).

In the present study the terrain effects on target detectability are largely ignored in an attempt to keep the problem manageable.

A FLIR sensor can operate in one of the following three field of view specifications at a time:

- pilotage field of view $39^\circ \times 30^\circ$ (wide field of view);
- medium field of view $6.2^\circ \times 4.6^\circ$;
- narrow field of view $2.1^\circ \times 1.6^\circ$.

An additional digital zoom is also available at field of view of $1.1^\circ \times 0.8^\circ$. The field of regard of FLIR is azimuth $\pm 120^\circ$ and elevation -25°, $+40^\circ$. Targets can be manually locked by the pilot using FLIR image and automatically tracked by the FLIR system while within the field of view. It is possible that if the pilot changes the field of view while tracking a target or targets, automatic tracking will fail and will have to be reset. In that case the pilot has to detect the targets again.

High detection accuracy and wide field of view are conflicting objectives. That is, the narrower field of view the higher detection accuracy. The general problem of intelligent scheduling of FLIR sensor for tracking of multiple targets is depicted in Fig. 1. The research problem addressed in this paper is the scheduling of a narrow Field-Of-View (FOV) sensor for tracking multiple targets of different types using reinforcement learning agents.

3. REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING

Reinforcement learning (RL) [2][5], or neuro-dynamic programming (NDP) [1], was originally developed as a way of describing observations in animal behaviour and may be considered an extension of dynamic programming [6][7].By describing a problem in terms of a Markov decision process (MDP), dynamic programming (DP) techniques may be applied to find an optimal solution. DP

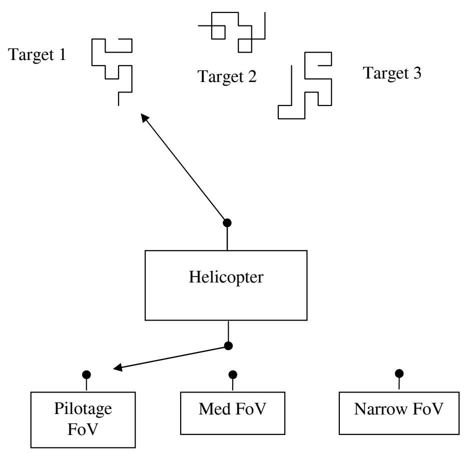

Figure 1: Target and field of view scheduling for optimal tracking.

however requires that a complete and accurate model of the environment is available, and this may not be the case. RL allows an agent to learn in an uncertain environment by building up an internal model of its environment through sample interactions. As its experience with the environment increases the learning agent may predict with more confidence which actions will tend to lead to preferred results.

3.1 Markov Decision Processes

Let us assume we wish an agent to learn an optimal course of choices to take in a given environment. The environment may be in one of a finite number of states, $s \in S$, and the agent may take one action from the set of admissible actions, $a \in A$. As a consequence of taking action a in state s two things occur: firstly, a scalar reward is received by the agent, $r(s,a)$, taken from a reward distribution $W(s,a)$, and secondly, the environment moves to a new state, s', dependent upon a set of transition probabilities:

$$Pr(s'\,|\,s,a),$$

where

$$\sum_{s'\in S} Pr(s'\,|\,s,a) = 1.$$

As the reward received and the transition to the new state s' are both dependent only on the previous state s and the action taken a, the system is 'memoryless', i.e., it has the Markovian property, and is described as a Markov decision process.

The optimality of the agent's quest for a course of action will be determined by a maximisation of received rewards, expressed over an infinite-horizon from time t as:

$$R_t = r_t + \sum_{i=1}^{\infty} \gamma^i r_{t+i}, \tag{1}$$

where γ, $0 < \gamma < 1$, is a discount parameter allowing a bounded sum over the infinite horizon, and r_t denotes the reward received at time t, dependent on s_t and a_t and drawn from the reward function:

$$r_t = W(s_t, a_t). \tag{2}$$

If the reward generated is stochastic in nature then its expected value must be used, i.e. $r_t = E\{W(s_t, a_t)\}$.

3.2 Dynamic Programming

Central to the idea of DP is policy evaluation, i.e. the determination of the relative value of a given policy. Specifically, the value of

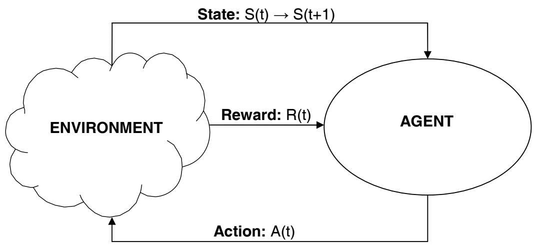

Figure 2: Reinforcement learning agent.

following a policy π from state s at time t, s_t, can be evaluated by the rewards received:

$$V^{\pi}(s_t) = W(s_t, \pi(s_t)) + \sum_{i=1}^{\infty} \gamma^i W(s_{t+i}, \pi(s_{t+i})), \tag{3}$$

where $\pi(s_t)$ denotes the action a taken in state s_t when following policy π. This may then be expressed recursively:

$$V^{\pi}(s_t) = W(s_t, \pi(s_t)) + \sum_{s_{t+1}\in S} Pr(s_{t+1}\,|\,s_t, \pi(s_t))\,\gamma V^{\pi}(s_{t+1}). \tag{4}$$

By repeatedly sweeping over the entire state-space an agent can update its estimated policy evaluations via equation (4), with each complete sweep garnering a more accurate estimation under a stationary environment. The process of completely updating the values of policies based upon all subsequent states is termed *full backup*.

The ability to evaluate policies enables the search for an optimal policy. The value of following the optimal policy, π^*, from state s can be expressed as the value of taking the optimal action in state s and thereafter following π^*, that is:

$$V^*(s) = \max_{a\in A} (\, W(s,a) + \sum_{s'\in S} Pr(s'\,|\,s,a)\,\gamma V^*(s') \,) \quad \text{for all } s \in S, \tag{5}$$

with the optimal policy itself being:

$$\pi^*(s) = \max_{a\in A} (\, W(s,a) + \sum_{s'\in S} Pr(s'\,|\,s,a)\,\gamma V^*(s') \,). \tag{6}$$

Equation (5), known as the Bellman optimality equation, allows an agent to find an optimal policy. If the value of first taking action a in state s and then following policy π is greater than strictly following policy π from state s, i.e., if

$$r(s,a) + \gamma V^{\pi}(s') > V^{\pi}(s), \tag{7}$$

then an improvement over policy π has been found. A policy that cannot be improved upon is an optimal policy, the existence of which is guaranteed for a discounted infinite-horizon system [8].

3.3 Reinforcement Learning

3.3.1 *Introduction to Reinforcement Learning*

Whilst DP is a powerful tool for solving problems characterised by MDPs, it assumes perfect knowledge of the environment. This may not be a realistic assumption in certain cases. RL deals with unavailability of a perfect environment model by gradually taking samples from an environment through interactions with it, over time building up approximations of both the reward and transition probabilities of the system. An imperfect environment model may be sufficient for the development of a usable sub-optimal solution.

Fig. 2 shows a simple RL scheme. The environment can be characterised by the configuration or values of a certain number of its features, which is called its state, denoted by $S(t)$ at time t in Fig. 2. Each state has an intrinsic value, dependent upon a certain immediate reward or cost, denoted by $R(t)$ at time t, which is generated when it is entered. At each discrete moment in time the agent may take one of a number of possible actions, $A(t)$, which affects the next state of the system, $S(t+1)$, and therefore the next reward/cost experienced, according to certain transition probabilities.

The agent's choice of action, given the current state of the system, is modified by experience, i.e., it uses its past experience of action taken in a certain system state and reward/cost experienced to update its decision making process for future actions. A policy of actions to be taken given particular system states is developed over time by the agent as it interacts with the environment. Alternative policies are evaluated in terms of the reward function. Each state is associated with a value function which is an approximation of the future rewards that may be expected starting from that particular state if an optimal policy was adhered to. As exploration of the problem by the learning agent proceeds, the values associated with particular states may be modified to be closer to the value of the state that preceded it.

3.3.2 *How Reinforcement Learning Agents Learn*

Let us assume a problem readily described by a MDP, but a perfect model of the environment is not available to the agent, i.e., neither the transition probabilities, $Pr(s' \mid s,a)$, nor the reward function, $W(s_t,a_t)$, are known. Furthermore, suppose that the reward function is stochastic in nature due to environment noise. If each possible environment state is sampled infinitely often with records kept of rewards garnered and state transitions experienced, then a model can be built of the environment provided it remains stationary. Specifically, the estimated value of a given state s at time t, assuming the agent takes its current estimated optimal action, may be evaluated by a *sample backup* as:

$$V_{t+1}(s) = V_t(s) + \alpha(r_t + \gamma V_t(s') - V_t(s)), \qquad (8)$$

where α, $0 < \alpha \leq 1$, is a learning rate parameter, often set to $1/x$ where x represents the number of times state s has been visited, and s' and a' are the next state visited and action taken respectively. If an environment is expected to be non-stationary α may be held constant in order to provide a tracking ability. This procedure of equation (8) allows an agent to update its estimated value by the difference between what it was expecting to receive and what it actually did, $r_t + \gamma V_t(s') - V_t(s)$, by making backups based upon sample interactions with the environment (albeit bootstrapping by using its current estimate of the next state encountered). This method of updating is termed *temporal difference learning*, with equation (8) being the update method used for Sutton's TD(0) algorithm [9].

The value of taking specific actions in a given state, i.e., the value of a given state-action pair, may also be estimated by extending equation (8). Assuming $Q_t(s,a)$ represents the learning agent's estimate at time t of the value of taking action a in state s then $Q_{t+1}(s,a)$ may be updated by:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha(r_t + \gamma Q_t(s',a') - Q_t(s,a)), \qquad (9)$$

for the SARSA algorithm [2]. SARSA converges to an optimal policy with probability 1 if all admissible state-action pairs are visited infinitely often and its policy converges to a greedy policy, given certain assumptions. This can be achieved for example by using an ε-greedy policy with $\varepsilon \to 0$ as $t \to \infty$ for both the estimation of the value of the next state-action pair and the decision of action to take at time t.

A process of updating the state-action value estimates for SARSA proceeds as follows. After initialisation, the agent retrieves its current action-value estimate of the previous state-action pair to occur and the current state-action pair, and the reward obtained immediately after the previous state-action pair was enacted. These three values are used along with the learning rate and discount parameters to update the agent's estimate of the previous state-action pair, as per equation (9). The current state-action pair is then stored as the previous state-action pair, and the reward obtained after its enactment is stored as the reward to be used in the next update procedure.

4. TARGET MODEL

The target movements are modelled as first-order discrete-time Markov processes. If target i is at location $s_i(t) = [x_i(t), y_i(t)]^T$ at discrete time $t \in \mathbb{Z}^+$, it will be at $s_i(t+1) = s_i(t) + \delta$ at time $t+1$ with probability $p_{ij}(t) = P(s_i(t+1)|s_i(t))$, $j = 0,1,\ldots,4$ where

$$\begin{aligned}
p_{i0}(t) &= P(\delta = 0) \quad \text{(target stationary)}\\
p_{i1}(t) &= P(\delta = [0,\delta_y]^T) \quad \text{(move up)}\\
p_{i2}(t) &= P(\delta = [0,-\delta_y]^T) \quad \text{(move down)}\\
p_{i3}(t) &= P(\delta = [\delta_x,0]^T) \quad \text{(move right)}\\
p_{i4}(t) &= P(\delta = [-\delta_x,0]^T) \quad \text{(move left).}
\end{aligned}$$

Here ^{T} denotes matrix/vector transpose. For target i, the transition probabilities at time t, $p_{ij}(t)$, are represented by the vector $p_i(t)$:

$$p_i(t) = [p_{i0}(t), \ldots, p_{i4}(t)]^T. \qquad (10)$$

The Markov processes are stochastic, i.e. $\sum_{j=0}^{4} p_{ij}(t) = 1 \; \forall i$.

In general each target has a unique transition probability vector. The transition probabilities are unknown by the helicopter. Thus part of the challenge in sensor scheduling is implicitly to estimate these probabilities by way of exploring. Any target that is scanned by the sensor on-board the helicopter has a probability, $\pi, 0 \leq \pi \leq 1$, of being destroyed immediately after scanning. This results in environment state transitions that have a dependence on actions taken.

Fig. 3 shows an example plot of target evolutions over 200 timesteps taken from a simulation run for nine targets with random transition probability vectors $p_1(t), \ldots, p_9(t)$ where

$$p_{ij}(t) = \frac{x_{ij}}{\sum_{k=0}^{4} x_{ik}}, \quad 0 \leq j \leq 4, \quad 1 \leq i \leq 9, \quad \forall t \qquad (11)$$

and x_{ij} is a uniformly distributed random variable over the unit interval $[0,1)$.

The field of view is assumed to be a rectangular grid with 251 points along both the x-axis and y-axis, with the x-axis ranging from -500 to 500 m and the y-axis from 0 to 1000 m. Targets that would move off the grid due to movement in a particular direction instead remain on the grid by being stationary. The helicopter is at the origin of the local coordinate system.

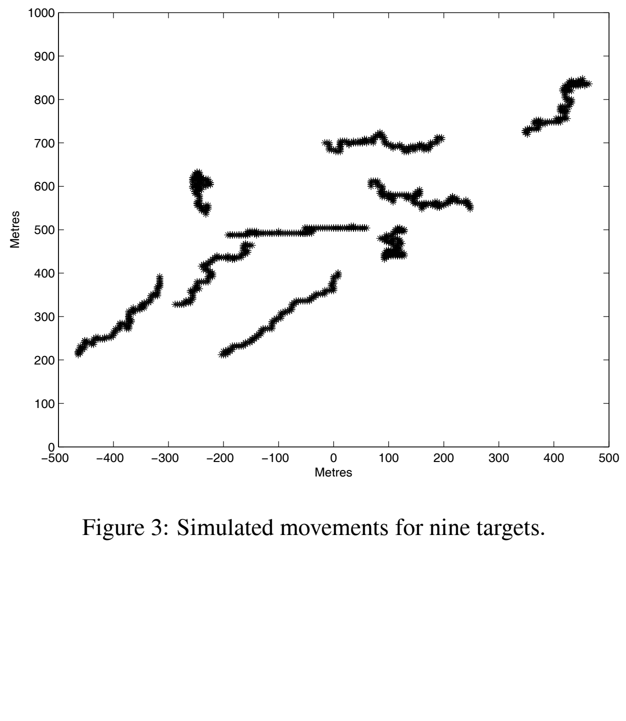

Figure 3: Simulated movements for nine targets.

5. RL SENSOR SCHEDULING

An RL based sensor scan optimisation scheme is proposed which adaptively chooses which of a number of available targets to scan at a given time instant. Each target i has an associated type, $y_i \in \{1,2,3,4\}$, representing the threat priority of the target, i.e. a target with a threat priority of 1 is considered more dangerous than a target with a threat priority of 4, all other factors being equal. It is considered that the most desirable target to scan is the target which currently poses the highest threat to the helicopter. In this paper, we consider that the threat level of a target is a combination of the distance from the target to the helicopter and the threat priority of the target.

The transition probabilities of targets are drawn from a random distribution. Targets are also given a bias to move towards the helicopter by setting:

$$\{p_{i1}(t), p_{i2}(t)\} = \begin{cases} \{p_{i2}(t), p_{i1}(t)\}, & \text{if } p_{i1}(t) > p_{i2}(t), \\ \{p_{i1}(t), p_{i2}(t)\}, & \text{otherwise.} \end{cases} \tag{12}$$

This formulation results in targets tending to move away from their original locations over time.

The reward obtained by the learning agent when scanning a target, denoted as c_t, is based upon the Euclidean distance of the scanned target from the helicopter and the target threat priority. Furthermore, this value is assumed to be noisy:

$$c_t = d_{it}\, y_i + \sigma_0\, g \sqrt{d_{it}}, \tag{13}$$

where d_{it} represents the Euclidean distance of target i at time t from the helicopter, y_i represents the scanned target's threat priority, σ_0 is a noise scaling factor, and g represents an additive Gaussian noise.

As the reward is partially based upon target distance, scanning the closer of two targets of the same threat priority will result in a lower reward in a noise-free environment. This allows the reformulation of the problem from a reward-maximisation to a cost-minimisation, i.e. the action with the *minimum* estimated cost associated with it is chosen in the case of a greedy action selection.

The update rule for the learning agent is:

$$Q_{t+1}(a) = Q_t(a) + \alpha(c_t + \gamma Q_t(a') - Q_t(a)), \tag{14}$$

where a represents the target chosen to be scanned at time t, and c_t represents the reward obtained at time t given action a. By removing any state component from the agent's action value estimation it is envisaged that the learning agent may be better able to rapidly learn optimal policies in a highly dynamic environment. A target that is destroyed is replaced by a new target on the grid, with a new random location and movement probabilities and a Q_t value of 0.

6. SIMULATION STUDIES

The proposed RL-based sensor scan optimisation scheme was evaluated via a number of computer simulations. The computer simulation environment consisted of a two-dimensional grid featuring 251 possible positions along each dimension. Nine mobile targets were simulated, with movements according to the scheme described in Section 4.

The RL-based scheme was compared with a round-robin scheme that chose available targets sequentially according to their target number, and a random scheme that chose targets randomly with equal probability. The distances of each unscanned target were unknown to the helicopter, thus it could only receive up-to-date yet noisy distance information of a given target by actively scanning it. Each trial was conducted for 1,000 timesteps, and the results of 100 trials were averaged and evaluated.

For the purposes of evaluation, at each action an instantaneous cost was incurred, equal to the true distance of the target scanned multiplied by the target threat priority. Therefore, an optimal policy, i.e. one with a minimum total cost, would scan the highest threat target at every time step. Given that the evolving movement of unscanned targets is unavailable to the helicopter, repeatedly choosing the initially closest target and never scanning other targets cannot be guaranteed to result in an optimal policy. Also, given the learning agent only has access to "noisy" distance measurements, repeated scanning of a stationary target over time will tend to give a more accurate measure of its true distance.

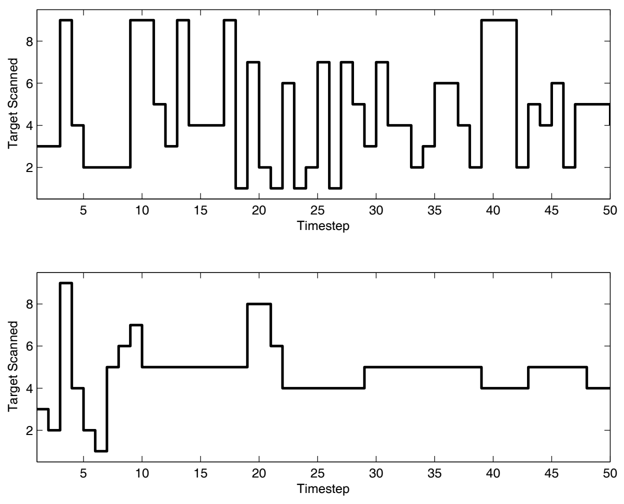

Figure 4: Comparison of uniform random and Gibbs exploratory action selection.

A ε-greedy action selection policy was implemented, where a greedy action was selected with a probability of $1-\varepsilon$. In the case of a non-greedy action, a Gibbs distribution was used to select which exploratory action to take as follows:

$$a_t = \frac{e^{-Q_t(a)/\tau}}{\sum_{b=1}^{i} e^{-Q_t(b)/\tau}}, \tag{15}$$

for i targets, with the temperature parameter τ held constant at $1/\varepsilon$. This results in the agent selecting lower cost non-greedy actions with a higher probability than higher cost non-greedy actions. The motivation for this implementation is to achieve a reduction in the cost of exploratory behaviour, at the expense of an even exploration policy. Given that as $\tau \to 0$ action selections are skewed towards greediness, it was decided to make the temperature inversely proportional to ε due to the fact that for lower ε exploratory actions should be less greedy than for higher ε. The rationale being, that for lower ε exploratory actions are taken less frequently, thus they should be more likely to choose previously high-cost actions, lest those actions not be sufficiently explored. Additionally, by setting the temperature to a function of ε only one exploration parameter needs to be chosen. Lastly, the temperature was chosen to remain stationary as the environment is dynamic in terms of target movement and presence and thus it cannot be conjectured that the agent's value estimates will converge to their true values. An illustration of the behaviour of the exploration scheme can be seen in Fig. 4, which shows a comparison between exploration using a uniform random distribution for exploratory action selection (upper plot) and utilising the Gibbs distribution for non-greedy action selection (lower plot) for $\varepsilon = 1$. This scheme is innovative in that it is conventional to use a Gibbs or Boltzmann distribution for *all* action selections in a softmax action selection architecture [2], not only for exploratory actions. This design allows a leveraging of the exploration parameter ε, and its constancy due to the dynamic nature of environment, to provide a lower-cost exploration policy without the need to select and maintain an additional agent parameter. To the best of our knowledge, this is the first time this particular type of exploration scheme has been used.

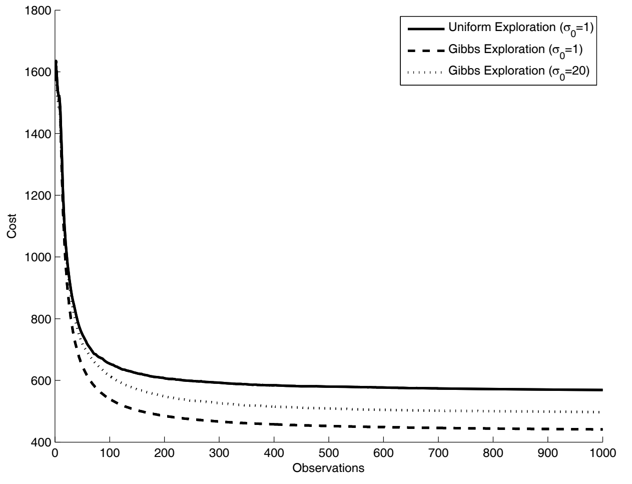

Figure 5: Sensor cost for Uniform Random and Gibbs exploration, for $\sigma_0 = 1$ and $\sigma_0 = 20$.

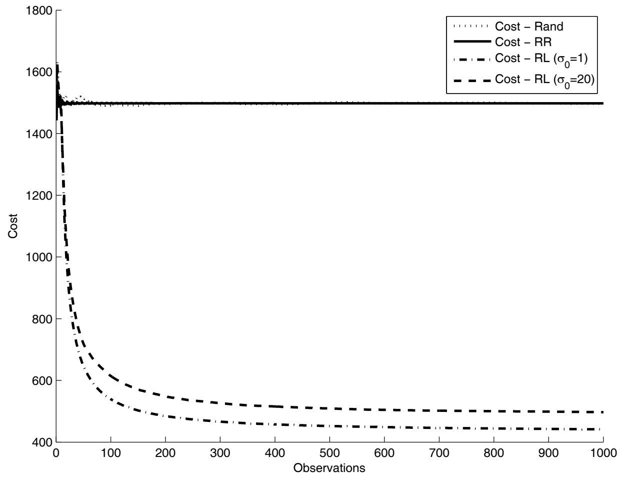

Figure 6: Sensor cost for Round Robin, Random, and RL for $\sigma_0 = 1$ and $\sigma_0 = 20$.

It can be noticed in Fig. 4 that the uniform random action selection (upper subplot) does not obviously favour a subset of the available actions, whilst the Gibbs action selection (lower subplot) favours scanning targets 4 and 5. By favouring low-cost exploratory actions not only can the cost of exploration be reduced, but actions which may be on the threshold of becoming optimal are likely to be more often explored, thereby possibly leading to a greater ability for the agent to track optimal actions throughout the scenario.

The Q values of all actions were initiated to 0 to encourage early exploration, as greedy action selection favours minimally-valued actions. The learning rate parameter, α, was set to 0.9, and the exploration and discount parameters, ε and γ respectively, were both set to 0.1. The probability of a target being destroyed immediately after being scanned, π, was set to 0.1.

The true costs of the simulated sensor scan scheduling schemes for 100 trials of 1,000 timesteps each is shown in Fig. 5. At each time step the mean cost of all previous scan actions is calculated, and then averaged for all trials. As can be seen, the Gibbs exploratory action selection results in a much lower cost after the initial phase of the scenarios for a noise factor of $\sigma_0 = 1$. Furthermore, even with a higher noise factor of $\sigma_0 = 20$, the Gibbs non-greedy action selection scheme still out-performs the uniform random action selection scheme with a noise factor of $\sigma_0 = 1$. Therefore, the benefit of the Gibbs exploratory action scheme significantly outweighs the overhead of much noisier data.

Next the RL-based sensor scan scheduling architecture was compared to random target selection round-robin schemes, as shown in Fig. 6. It may be seen in the figure that the costs of employing either a random target selection scheme (labelled 'RAND') or a round-robin scheme (labelled 'RR') are comparable. This is to be expected as neither scheme takes neither target distance nor type into account and over time the random scheme is expected to scan each available target similar number of times due to its action choice being drawn from a uniformly distributed random function. The RL-based sensor scan optimisation scheme exhibits a much lower cost over the vast majority of the time period for a noise factor $\sigma_0 = 1$ once the agent has learnt which targets pose the highest threat levels. Furthermore, even for a large noise factor ($\sigma_0 = 20$) the learning agent is readily able to determine a lower-cost scanning policy based upon target threat levels. This shows the learning agent has an appreciable robustness to noise, by sampling the potential targets a number of times it gains a more accurate estimation of each target's true threat level. It should be pointed out that the learning agent learns its policy in a fully on-line manner, i.e. there is no initial off-line learning period. The lack of an off-line training period manifests two benefits: the learning agent learns independent of any training data and thus is unaffected by any inaccuracy that training data may possess, and the learning agent is able to track action value estimates and therefore adjust its policies in a dynamic environment.

7. CONCLUSION

A novel RL-based dynamic sensor scan optimisation scheme has been presented in this paper that prioritises individual targets based upon an ongoing estimate of their threat level. The presented RL architecture is designed to learn in a fully on-line manner, i.e. without any off-line training period, and to be able to formulate an optimal or near-optimal policy in a highly dynamic environment using only 'noisy' data. The proposed scheme has been shown via extensive computer simulations to result in a greatly reduced sensor cost when compared to round robin and random target selection schemes, and to exhibit an appreciable robustness to noisy data measurements. It has also been shown that an innovative exploration policy utilising a Gibbs distribution to select non-greedy actions was able to improve performance when compared to a more conventional scheme of selecting exploratory actions via a uniform random distribution.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1999.

[3] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley, 1989.

[4] V. Krishnamurthy and R. J. Evans, "Hidden Markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking," *IEEE Trans. Signal Processing*, vol. 49, no. 12, pp. 2893–2908, December 2001.

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. 1.

[7] ——, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. 2.

[8] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[9] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.