Wireless Netw
DOI 10.1007/s11276-014-0762-6

Two timescale convergent Q-learning for sleep-scheduling in wireless sensor networks

L. A. Prashanth · Abhranil Chatterjee · Shalabh Bhatnagar

Abstract In this paper, we consider an intrusion detection application for Wireless Sensor Networks. We study the problem of scheduling the sleep times of the individual sensors, where the objective is to maximize the network lifetime while keeping the tracking error to a minimum. We formulate this problem as a partially-observable Markov decision process (POMDP) with continuous state-action spaces, in a manner similar to Fuemmeler and Veeravalli (IEEE Trans Signal Process 56(5), 2091–2101, 2008). However, unlike their formulation, we consider infinite horizon discounted and average cost objectives as performance criteria. For each criterion, we propose a convergent on-policy Q-learning algorithm that operates on two timescales, while employing function approximation. Feature-based representations and function approximation is necessary to handle the curse of dimensionality associated with the underlying POMDP. Our proposed algorithm incorporates a policy gradient update using a one-simulation simultaneous perturbation stochastic approximation estimate on the faster timescale, while the Q-value parameter (arising from a linear function approximation architecture for the Q-values) is updated in an on-policy temporal difference algorithm-like fashion on the slower timescale. The feature selection scheme employed in each of our algorithms manages the energy and tracking components in a manner that assists the search for the optimal sleep-scheduling policy. For the sake of comparison, in both discounted and average settings, we also develop a function approximation analogue of the Q-learning algorithm. This algorithm, unlike the two-timescale variant, does not possess theoretical convergence guarantees. Finally, we also adapt our algorithms to include a stochastic iterative estimation scheme for the intruder's mobility model and this is useful in settings where the latter is not known. Our simulation results on a synthetic 2-dimensional network setting suggest that our algorithms result in better tracking accuracy at the cost of only a few additional sensors, in comparison to a recent prior work.

Keywords Sensor Networks · Sleep-Wake scheduling · Reinforcement learning · Q-learning · Function approximation · Simultaneous perturbation · SPSA

Electronic supplementary material The online version of this article (doi:10.1007/s11276-014-0762-6) contains supplementary material, which is available to authorized users.

L. A. Prashanth (✉)
INRIA Lille, Nord Europe, Team SequeL, Villeneuve d'Ascq, France
e-mail: prashanth.la@inria.fr

A. Chatterjee
System Sciences and Automation, Indian Institute of Science, Bangalore, India
e-mail: abhranilc@ee.iisc.ernet.in

S. Bhatnagar
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
e-mail: shalabh@csa.iisc.ernet.in

1 Introduction

Considering the potential range of applications and the low deployment and maintenance cost, a lot of research attention has gone into the design of Wireless Sensor Networks (WSNs). In this paper, we investigate the use of WSNs for an intrusion detection application. In particular, we study the problem of scheduling the sleep times of the individual sensors, where the objective is to maximize the network lifetime while keeping the tracking error to a minimum.

Published online: 25 June 2014

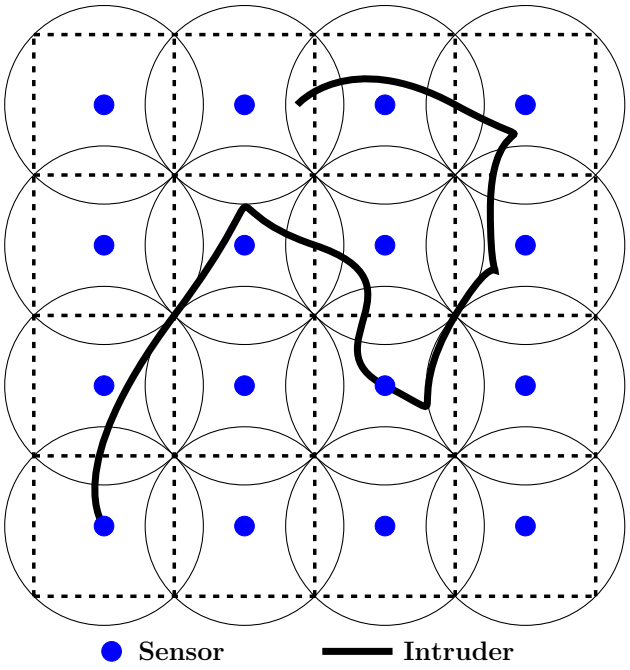

Fig. 1 Field of sensors and the movement of intruder considered

As illustrated in Fig. 1, we consider a centralized control setting for a sensor network involving N sensors and assume for simplicity that the sensors fully cover the area of interest. Each sensor can be either awake (i.e., active) or asleep. The control center collects sensing information periodically and then decides on the sleeping policy for the sensors. The location of the intruder at any instant can be any one of the N cells corresponding to the N sensors. The intruder movement is described by a Markov chain with a probability transition matrix P of size $N \times N$. Each entry $P_{ij} \in [0, 1]$ of the matrix P specifies the probability of the intruder moving from location i to j. The state of this Markov chain is the current location of the intruder to within the accuracy of a sensing region. The challenge is to balance the conflicting objectives of minimizing the number of sensors awake to reduce the energy cost, while at the same time having enough number of sensors awake to ensure a good tracking accuracy.

We formulate this problem as a partially-observable Markov decision process (POMDP), in a manner similar to [14]. However, unlike their formulation, we consider infinite horizon discounted and average cost objectives as performance criteria. The rationale behind the average cost objective is to understand the steady-state system behavior, whereas the discounted cost objective is more suitable for studying the transient behavior of the system.

MDPs [4] (and POMDPs) are useful frameworks for modeling real-time control problems such as the sleep scheduling that we consider in this paper. However, in practice, the transition dynamics of the MDP is unavailable and reinforcement learning (RL) approaches provide an efficient alternative. RL comprises of simulation-based sample-path techniques that converge to a good-enough policy in the long run. The reader is referred to [5, 31] for a comprehensive (text book) introduction to RL.

We base our solution approach on reinforcement learning (RL) formalisms, with specific emphasis on developing Q-learning [33] type algorithms for learning the optimal sleeping policy for the sensors. At the same time, to be computationally efficient, we employ linear approximation architectures. Linear function approximation allows us to handle the curse of dimensionality associated with high-dimensional state spaces (as is the case with the sleep-wake scheduling problem considered in this paper). To the best of our knowledge, RL with function approximation for sleep scheduling in WSNs has not been considered previously in the literature.

However, for problems involving high-dimensional state spaces, the Q-learning algorithm with function approximation may diverge or may show large oscillations, [2]. This is primarily due to the inherent nonlinearity in the Q-learning update rule resulting from the explicit maximization/minimization in the update procedure. To alleviate this problem, we propose a two-timescale Q-learning algorithm, borrowing the principle of using a simultaneous perturbation method for policy gradient estimation from a closely related algorithm for the discounted setting proposed in [6].

Our algorithm, proposed for continuous state-action spaces and the long-run average cost criterion, operates on two timescales and works on the simultaneous perturbation principle [9, 30]. In particular, after parameterizing the policy in a continuous space, the algorithm updates the policy parameter along the negative gradient direction using a well-known simultaneous perturbation method called simultaneous perturbation stochastic approximation (SPSA) [9, Chapter 5]. In particular, we employ a one-simulation SPSA estimate on the faster timescale for obtaining the policy gradients. On the other hand, along the slower timescale an on-policy TD-like update is performed for the Q-value parameters. This timescale separation together with the policy gradient estimation using SPSA gets rid of the off-policy problem present in vanilla Q-learning with function approximation. The resulting algorithm turns out to be a stochastic approximation scheme on the faster timescale, but a stochastic recursive inclusion [10, Chapter 5] scheme on the slower timescale. We provide a sketch of the convergence of this algorithm, with the detailed proof being available in an appendix to this paper. To the best of our knowledge, a convergent Q-learning type algorithm with function approximation to optimize a long-run average cost criterion in a POMDP with continuous state-action spaces (as is the case with the sleep-scheduling POMDP considered), has not been proposed earlier in the literature.

We summarize our contributions as follows:[1]

(i) In the average cost setting, we propose a novel two-timescale algorithm that performs on-policy Q-learning while employing function approximation. This algorithm is efficient owing to linear function approximators and possesses theoretical convergence guarantees. For the sake of comparison, we also develop a function approximation analogue of the Q-learning algorithm. This algorithm, unlike the two-timescale variant, does not possess theoretical convergence guarantees. The feature selection scheme employed in each of our algorithms manages the energy and tracking components in a manner that assists the search for the optimal sleep-scheduling policy.

(ii) In the discounted setting, we adapt the recently proposed two-timescale (convergent) variant of the Q-learning algorithm [6], with function approximation. Further, for the sake of comparison, we also develop a sleep-scheduling algorithm based on Q-learning with linear function approximation. These algorithms can be seen to be the discounted-cost counterparts of the algorithms described above for the average cost setting.

(iii) We also adapt our algorithms to a setting where the mobility model of the intruder is not available. We develop a stochastic iterative scheme that estimates the mobility model and combine this estimation procedure with the average cost algorithms mentioned above using multi-timescale stochastic approximation.

(iv) We validate our algorithms on a two-dimensional network setting, while also comparing their performance with the Q_{MDP} and FCR algorithms from [14]. Our algorithms are seen to be easily implementable, converge rapidly with a short (initial) transient period and provide more consistent results than the Q_{MDP} and FCR algorithms. Further, we observe that the procedure for estimating the mobility model of the intruder converges empirically to the true model.

2 Related work

Sleep scheduling is broadly related to the problem of resource allocation in wireless networks. A comprehensive survey of solution approaches, including RL-like schemes, is available in [11]. Further, considering this problem from a strategic, i.e., game-theoretic, perspective, the authors in [13] propose an auction based best-response algorithm. A two-timescale stochastic approximation algorithm for downlink scheduling in a cellular wireless system is proposed in [12].

In [27], the authors formulate an MDP model for intrusion detection and present algorithms to control the number of sensors in the wake state. In [18, 22, 23], the authors propose RL based medium access control (MAC) protocols for WSNs. The algorithms proposed there attempt to maximize the throughput while being energy efficient. In [22, 23], the authors propose Q-learning based algorithms, whereas, in [18], the authors propose an algorithm based on SARSA. In [16], the authors present two sleep scheduling algorithms for single object tracking. In [17], a sleep scheduling algorithm based on the target's moving direction has been proposed. In [19], the authors present a heuristic algorithm that uses dynamic clustering of sensors to balance energy cost and tracking error. In [3], the problem of finding an efficient sleep-wake policy for the sensors while maintaining good tracking accuracy by solving an MDP has been studied. In [20], the authors propose a Q-learning based algorithm for sleep scheduling. In [14, 15], the authors propose a POMDP model for sleep-scheduling in an object tracking application and propose several algorithms based on traditional dynamic programming approaches to solve this problem.

In comparison to previous works, we would like to point out the following:

(i) Some of the previously proposed algorithms, for instance [27], require the knowledge of a system model and this may not be available in practice. On the other hand, our algorithms use simulation-based values and optimize along the sample path, without necessitating a system model. (ii) Some algorithms, for instance [16], work under the waking channel assumption, i.e., a setting where the central controller can communicate with a sensor that is in the sleep state. Our algorithm do not operate under such an assumption. (iii) In comparison to the RL based approaches [18, 22, 23] for transmission scheduling at the MAC layer, we would like to point out that the algorithms proposed there (a) employ full state representations; (b) consider discrete state-action spaces (except [23] which adapts Q-learning for continuous actions, albeit with a discrete state space); (c) consider an MDP with perfect information, i.e., a setting where the states are fully observable; (d) consider only a discounted setting, which is not amenable for studying steady state system behaviour; (e) are primarily concerned with managing transmission in an energy-efficient manner and not with tracking an intruder with high-accuracy. In other words, the algorithms of [18, 22, 23] are not applicable in our setting as we consider a partially observable MDP with continuous state-action spaces, and

[1] A short version of this paper containing only the average cost setting and algorithms and with no proofs is available in [24]. The current paper includes in addition: (i) algorithms for the discounted cost setting; (ii) a detailed proof of convergence of the average cost algorithm using theory of stochastic recursive inclusions; and (iii) detailed numerical experiments.

with the aim of minimizing a certain long-term average cost criterion that involves the conflicting objectives of reducing energy consumption and maintaining a high tracking accuracy. (iv) Many RL based approaches proposed earlier for sleep scheduling (see [18, 22, 23, 29]) employ full state representations and hence, they are not scalable to larger networks owing to the curse of dimensionality. We employ efficient linear approximators to alleviate this. (v) While the individual agents in [13] employ a RL-based bidding scheme, their algorithm is shown to work well only empirically and no theoretical guarantees are provided. This is also the case with many of the earlier works on sleep scheduling/power management in WSNs using RL and this is unlike our two-timescale on-policy Q-learning based scheme that possesses theoretical guarantees. (vi) In [12], the authors derive an equivalent Bellman Equation (BE) after reducing the state space and establish convergence of their algorithm to the fixed point of the equivalent BE. However, there is no straightforward reduction of state space in our sleep scheduling problem and we employ efficient linear function approximators to alleviate the curse of dimensionality associated with large state spaces. (vii) In comparison to [14], which is the closest related work, we would like to remark that the algorithms proposed there, for instance, Q_{MDP} , attempt to solve a balance equation for the total cost in an approximate fashion at each time instant and no information about the solution thus obtained is carried forward to the future instants. Moreover, unlike [14], we consider long-run performance objectives that enable us to study both the transient as well as steady state system behavior.

In general, we would like to remark that unlike previous works on sleep-scheduling, we propose RL-based algorithms that observe the samples of a cost function from simulation and through incremental updates find a 'good enough' policy that minimizes the long-run (average or discounted) sum of this cost. The term 'good enough' here refers to the solution of a balance equation for the long term costs, where function approximation is employed to handle the curse of dimensionality. Our algorithms are simple, efficient and in the case of the two-timescale on-policy Q-learning based schemes, also provably convergent.

3 POMDP formulation

The state s_k at instant k for our problem is $s_k = (l_k, r_k)$, where $r_k = (r_k(1), \ldots, r_k(N))$, is the vector of residual (or remaining) sleep times, with $r_k(i)$ denoting the residual sleep time of sensor i at time instant k. Further, l_k refers to the location of the object at instant k and can take values $1, \ldots, N$. The residual sleep time vector r_k evolves as follows: $\forall i = 1, \ldots, N$,

$$r_{k+1}(i) = (r_k(i) - 1)\mathcal{I}_{\{r_k(i) > 0\}} + a_k(i)\mathcal{I}_{\{r_k(i)=0\}}. \quad (1)$$

In the above $\mathcal{I}_{\{C\}}$ denotes the indicator function, having the value 1 when the condition C is true and 0 otherwise. The first term in (1) indicates that the residual sleep time is decremented by 1 if sensor i is in sleep state, while the second term expresses that if sensor i is in wake state, it is assigned a sleep time of $a_k(i)$. Here $a_k = (a_k(1), \ldots, a_k(N))$ denotes the chosen sleep configuration of the N sensors at instant k.

The single-stage cost function has two components - an energy cost for sensors in the wake state and a tracking cost. We use an energy cost $c \in (0, 1)$ for each sensor that is awake and a tracking cost of 1 if the intruder location is unknown. Let S_k denote the set of indices of sensors that are in sleep state. Then the single-stage cost $g(s_k, a_k)$ at instant k has the form,

$$g(s_k, a_k) = \sum_{\{i: r_k(i)=0\}} c + \mathcal{I}_{\{r_k(l_k) > 0\}}. \quad (2)$$

Since the number of sensors is finite, the single-stage cost is uniformly bounded. The algorithms that we design subsequently find the optimal strategy for minimizing the single-stage cost (2) in the long-run average cost sense. Note that, unlike the formulation in [14], we do not consider a special termination state which indicates that the intruder has left the system.[2]

The states, actions and single-stage cost function together constitute an MDP. However, since it is not possible to track the intruder at each time instant (i.e., l_k is not known for all k) as the sensors at the location from where the intruder passes at a given time instant may be in the sleep state, the problem falls under the realm of MDPs with imperfect state information, or alternatively partially observed MDP (POMDP). Following the notation from [14], the observation z_k available to the control center is given by $z_k = (s_k, o_k)$, where s_k is as before and $o_k = l_k$ if the intruder location is known, or a special value ζ otherwise. Thus, the total information available to the control center at instant k is given by $I_k = (z_0, \ldots, z_k, a_0, \ldots, a_{k-1})$, where I_0 denotes the initial state of the system. The action a_k specifies the chosen sleep configuration of the n sensors and is a function of I_k. As pointed out in [14], in the above POMDP setting, a sufficient statistic is $\hat{s}_k = (p_k, r_k)$, where $p_k = P(l_k|I_k)$ and r_k is the remaining sleep time mentioned above. Note that $p_k = (p_k(1), \cdots, p_k(N))$ is the distribution

[2] Since we study long-run average sum of (2) (see (4) below), we can consider the problem of tracking an intruder in an infinite horizon, whereas a termination state in [14] was made necessary as they considered a total cost objective.

at time step k of the object being in one of the locations $1, 2, \cdots, N$ and evolves according to

$$p_{k+1} = e_{l_{k+1}} \mathcal{I}_{\{r_{k+1}(l_{k+1})=0\}} + p_k P \mathcal{I}_{\{r_{k+1}(l_{k+1})>0\}}, \tag{3}$$

where e_i denotes an N-dimensional unit vector with 1 in the ith position and 0 elsewhere. The idea behind the evolution of p_k is as follows:

(i) The first term refers to the case when the location of the intruder is known, i.e., the sensor at l_{k+1} is in the wake state;

(ii) the second term refers to the case when intruder's location is not known and hence, the intruder transitions to the next distribution p_{k+1} from the current p_k via the transition probability matrix P.

Note that the evolution of p_k in our setting differs from [14], as we do not have the termination state. With an abuse of terminology, henceforth we shall refer to the sufficient statistic $\hat{s}_k$ as the state vector in the algorithms we propose next. Further, we would like to emphasize here that our algorithms do not require full observation of the state vector. Instead, by an intelligent choice of features that rely only on p_k, the algorithms obtain a sleeping policy that works well.

4 Average cost setting

The long-run average cost $J(\pi)$ for a given policy π is defined as follows:

$$J(\pi) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} g(s_n, a_n), \tag{4}$$

starting from any given state i (i.e., with $s_0 = i$). In the above, the policy $\pi = \{\pi_0, \pi_1, \pi_2, \ldots\}$ with π_n governing the choice of action a_n at each instant n.

The aim here is to find a policy $\pi^* = \text{argmin}_{\pi\in\Pi} J(\pi)$, where Π is the set of all admissible policies. A policy π is admissible if it suggests a feasible action at each time instant n.

Let $h(x)$ be the differential cost function corresponding to state x, under policy π. Then,

$$h(x) = \sum_{n=1}^{\infty} E[g(s_n, a_n) - J(\pi) | s_0 = x, \pi], \tag{5}$$

is the expected sum of the differences between the single-stage cost and the average cost under policy π when $x \in \mathcal{S}$ is the initial state. Let $J^* = \min_{\pi\in\Pi} J(\pi) \triangleq J(\pi^*)$ denote the optimal average cost and let h^* denote the optimal differential cost function corresponding to the policy π^*. Then, $(J^*, h^*(x)), x \in \mathcal{S}$ satisfy the following Bellman equation (see [28]):

$$J^* + h^*(x) = \min_a (g(x,a) + \int p(x,a,dy) h^*(y)), \forall x \in \mathcal{S}, \tag{6}$$

where $p(x, a, dy)$ denotes the transition probability kernel of the underlying MDP. Now, define the optimal Q-factors $Q^*(x,a), x \in \mathcal{S}, a \in \mathcal{A}(x)$ as

$$Q^*(x,a) = g(x,a) + \int p(x,a,dy) h^*(y). \tag{7}$$

From (6) and (7), we have

$$J^* + h^*(x) = \min_a Q^*(x,a), \forall x \in \mathcal{S}. \tag{8}$$

Now from (7) and (8), we have

$$Q^*(x,a) = g(x,a) + \int p(x,a,dy)(\min_b Q^*(y,b) - J^*) \text{ or}$$
$$J^* + Q^*(x,a) = g(x,a) + \int p(x,a,dy) \min_b Q^*(y,b), \tag{9}$$

for all $x \in \mathcal{S}, a \in \mathcal{A}(x)$. An advantage with (9) is that it is amenable to stochastic approximation because the minimization is now (unlike (6)) inside the conditional expectation. However, in order to solve (9), one requires knowledge of the transition kernel $p(x, a, dy)$ that constitutes the system model. Moreover, one requires the state and action spaces to be manageable in size. The algorithms presented subsequently work under lack of knowledge about the system model and further, are able to effectively handle large state and action spaces by incorporating feature based representations and function approximation.

For the two-timescale on-policy Q-learning scheme (TQSA-A), we consider a parameterized set of policies that satisfy the following assumption:

Assumption 1 For any state-action pair (x, a), $\pi_w(x, a)$ is continuously differentiable in the parameter w.

The above is a standard assumption in policy gradient RL algorithms (cf. [8]). A commonly used class of distributions that satisfy this assumption for the policy π is the parameterized Boltzmann family, where the distributions have the form

$$\pi_w(x,a) = \frac{e^{w^\top \sigma_{x,a}}}{\sum_{a' \in A(x)} e^{w^\top \sigma_{x,a'}}}, \forall x \in \mathcal{S}\ ,\ \forall a \in \mathcal{A}(x). \tag{10}$$

In the above, the parameter $w = (w_1, \ldots, w_N)^T$ is assumed to take values in a compact and convex set $C \subset \mathbb{R}^N$.

Before we proceed further, it is important to note there that in our setting, we have a continuous state-action space. Hence, to implement our Q-learning algorithms, we discretize the space to a finite grid (as is commonly done in practice). In what follows, we shall consider $(x, a), (y, b)$ to

take values on the aforementioned finite grid of points and $p(x,a,y)$ to denote the transition probabilities of the resulting Markov chain.

5 Average cost algorithms

For the ease of exposition, we first describe the Q-learning algorithm that uses full-state representations. Next, we discuss the difficulty in using this algorithm on a high-dimensional state space (as is the case with the sleep-wake control MDP) and subsequently present our average cost algorithms that employ feature based representations and function approximation to handle the curse of dimensionality.

5.1 Q-learning with full state representation

This algorithm, proposed in [1], is based on the relative Q-value iteration (RQVI) procedure. Let s_{n+1} denote the state of the system at instant $(n+1)$ when the state at instant n is x and action chosen is a. Let $Q_n(x,a)$ denote the Q-value estimate at instant n associated with the tuple (x,a). The RQVI scheme (assuming a finite number of state-action tuples)

$$Q_{n+1}(x,a) = g(x,a) + \sum_y p(x,a,y) \underbrace{\min_{b\in\mathcal{A}(y)} Q_n(y,b)}_{(\mathbf{I})} - \underbrace{\min_{r\in\mathcal{A}(s)} Q_n(s,r)}_{(\mathbf{II})}, \tag{11}$$

where $s \in S$ is a prescribed (arbitrarily chosen) state.[3] Note, unlike the value iteration scheme for discounted MDPs, the recursion (11) includes an additional term (see (II) in (11)). This term arises due to the nature of the Bellman equation for average cost MDPs (see (9)) that also contains the optimal average cost J^*. Here the state s can be arbitrarily chosen because one is interested in estimating not just the average cost, but also the differential cost function. This results in solving a system of $(n+1)$ unknowns using n equations. In order to make this system feasible, one fixes the differential cost for one of the (arbitrarily chosen) state to be a fixed value and then solves for the remaining n values using the system of n equations. It has been shown in [1] that term (II) in (11) converges to J^* and term (I) in (11) converges to the optimal differential cost function $h^*(\cdot)$.

[3] A simple rule to choose a state s such that there is a positive probability of the underlying MDP visiting s. Such a criterion ensures that the term (II) of (11) converges to the optimal average cost J^*.

The Q-learning algorithm for the average cost setting estimates the 'Q-factors' $Q(x,a)$ of all feasible state-action tuples (x,a), i.e., those with $x \in S$ and $a \in \mathcal{A}(x)$ using the stochastic approximation version of (11). The update rule for this algorithm is given by

$$Q_{n+1}(x,a) = Q_n(x,a) + a(n)(g(x,a) + \min_{b\in A(y)} Q_n(y,b) - \min_{r\in A(s)} Q_n(s,r)), \tag{12}$$

for all $x \in S$ and $a \in A(x)$. In the above, y is the simulated next state after x when action a is chosen in state x and $a(n), n \geq 0$ are the step-sizes that satisfy the standard stochastic approximation conditions, i.e., $\sum_n a(n) = \infty$ and $\sum_n a(n)^2 < \infty$. The last term $\min_{r\in A(s)} Q_n(s,r)$ in (12) asymptotically converges to the optimal average cost per stage. Further, the iterates in (12) converge to the optimal Q-values $Q^*(i,a)$ that satisfy the corresponding Bellman equation (9) and $\min_{a\in A(i)} Q_n(i,a)$ gives the optimal differential cost $h^*(i)$. The optimal action in state i corresponds to $\text{argmin}_{a\in A(i)} Q^*(i,a)$.

5.2 Need for function approximation

While Q-learning does not require knowledge of the system model, it does suffer from the computational problems associated with large state and action spaces as it stores the $Q(s,a)$ values in a look-up table and requires updates of all $Q(s,a)$ values at each step for convergence. In our setting, this algorithm becomes intractable as the state-action space becomes very large. Even when we quantize probabilities as multiples of 0.01, and with 7 sensors, the cardinality of the state-action space $|S \times A(S)|$ is approximately $100^8 \times 4^7 \times 4^7$ if we use an upper bound of 3 for the sleep time alloted to any sensor. The situation gets aggravated when we consider larger sensing regions (with corresponding higher number of sensors). To deal with this problem of the curse of dimensionality, we develop a feature based Q-learning algorithm as in [26]. While the full state Q-learning algorithm in (12) cannot be used on even moderately sized sensing regions, its function approximation based variant can be used over larger network settings.

5.3 Algorithm structure

Both our algorithms parameterize the Q-function using a linear approximation architecture as follows:

$$Q(s,a) \approx \theta^T \sigma_{s,a}, \quad \forall s \in S, a \in A(s). \tag{13}$$

In the above, $\sigma_{s,a}$ is a given d-dimensional feature vector associated with the state-action tuple (s,a), where

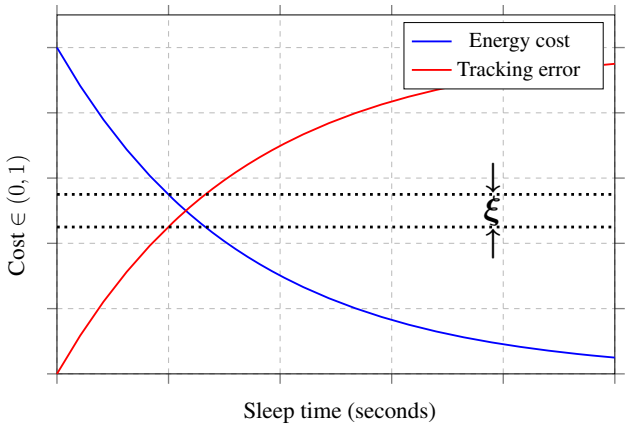

Fig. 2 Idea behind the feature selection scheme

$d << |S \times A(S)|$ and θ is a tunable d-dimensional parameter. The Q-value parameter $\theta = (\theta_1, \ldots, \theta_d)^T$ is assumed to take values in a compact and convex set $D \subset \mathbb{R}^d$.

Our algorithms are online, incremental and obtain the sleeping policy by sampling from a trajectory of the system. After observing a simulated sample of the single-stage cost, the parameter θ is updated in the negative descent direction in both our algorithms as follows:

$$\theta_{n+1} = \Gamma(\theta_n - a(n)\sigma_{s_n,a_n} m_n), \tag{14}$$

where m_n is an algorithm-specific magnitude term and Γ is a projection operator that keeps the parameter θ bounded (a crucial requirement towards ensuring convergence of the scheme). Further, $a(n)$ are the step-sizes that satisfy standard stochastic approximation conditions. Note that $\nabla_\theta Q(s,a) = \sigma_{s,a}$ and hence (14) updates the parameter θ in the negative descent direction. The overall structure of our algorithms is given in Algorithm 1.

to select only those actions that ensure that the energy cost is ξ-close to the tracking error and then, among the ξ-optimal actions, selecting an action that minimizes the approximate Q-value.

Formally, the choice of features is given by

$$\sigma_{s_n,a_n} = (\sigma_{s_n,a_n}(1), \cdots, \sigma_{s_n,a_n}(N))^T, \tag{15}$$

where $\sigma_{s_n,a_n}(i), i \leq N$ is the feature value corresponding to sensor i. These values are defined as follows:

$$\delta_n^{a_n(i)} = \underbrace{\frac{1}{(a_n(i)+1)}}_{\textbf{energycost}} - \underbrace{\frac{\sum_{j=1}^{a_n(i)} [pP^j]_i}{\sum_{j=1}^{\infty} [pP^j]_i}}_{\textbf{trackingerror}}, \tag{16}$$

$$\sigma_{s_n,a_n}(i) = \begin{cases} \delta_n^{a_n(i)} & \text{if} \quad 0 \leq |\delta_n^{a_n(i)}| \leq \xi, \\ \top & \text{otherwise.} \end{cases} \tag{17}$$

In the above, $\top$ is a fixed large constant used to prune out the actions that are not ξ-close. The above choice of features involve pruning of actions, which is explained as follows: Consider an action $a_n(i)$ for the sensor i at time instant n. The sum of probabilities that the intruder will be at location i, over time instants $1, \ldots, a_n(i)$ is a measure of the tracking error. On the other hand, the energy saved by having sensor i sleep for $a_n(i)$ time units is proportional to $\frac{c}{a_n(i)+1}$. As illustrated in Fig. 2, the tracking error increases with the sleep time (dictated by the choice of $a_n(i)$), while the energy cost decreases. Thus, $\delta_n^{a_n(i)}$ measures the distance between the energy cost and tracking errors. Next, as illustrated with the two-dashed lines in Fig. 2, we now consider all those actions $a_n(i)$ such that the above two components are within ξ distance of each other (i.e., $|\delta_n^{a_n(i)}| \leq \xi$) and set the feature value σ_{s_n,a_n} to the above difference. On the other hand, for those actions that are outside the ξ-boundary, we set σ_{s_n,a_n} to a large constant, which ensures they are not selected.

Algorithm 1 Structure of our algorithms

1: **Initialization:** policy parameter $\theta = \theta_0$; initial state s_0
2: **for** $n = 0, 1, 2, \ldots$ **do**
3: Take action a_n based on a (algorithm-specific) policy depending on θ_n.
4: Observe the single-stage cost $g(s_n, a_n)$ and the next state s_{n+1}.
5: Update θ_{n+1} in a algorithm-specific manner.
6: **end for**
7: **return** Q-value parameter θ, policy parameter w.

5.4 Feature selection

The idea behind the feature selection scheme is to select an energy-efficient sleep configuration, i.e., a configuration that keeps as many sensors in the wake state as possible to track the intruder while at the same time has minimal energy cost. This is done by first pruning the actions so as

In the following section, we present the QSA-A algorithm for sleep-wake scheduling and subsequently present the second algorithm (TQSA-A). The latter algorithm (TQSA-A) is a convergent algorithm, unlike QSA-A.

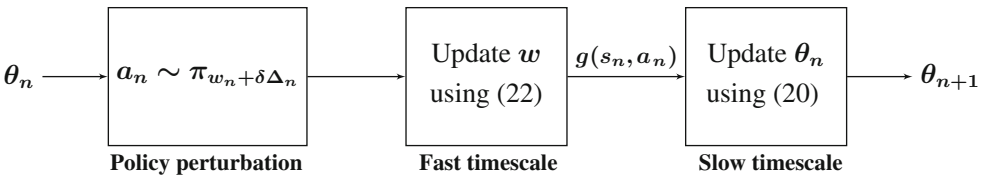

Fig. 3 Overall flow of the TQSA-A algorithm

5.5 Q-learning based sleep–wake algorithm (QSA-A)

This is the function approximation analogue of the Q- learning with average cost algorithm [1]. Let s_n, s_{n+1} denote the state at instants $n, n+1$, respectively, measured online. Let θ_n be the estimate of the parameter θ at instant n. Let s be any fixed state in S. The algorithm QSA-A uses the following update rule:

$$\theta_{n+1} = \theta_n + a(n)\sigma_{s_n,a_n}\Bigg(g(s_n,a_n) + \underbrace{\min_{v\in A(s_{n+1})} \theta_n^T \sigma_{s_{n+1},v}}_{\textbf{(I)}} - \underbrace{\min_{r\in A(s)} \theta_n^T \sigma_{s,r}}_{\textbf{(II)}} - \theta_n^T \sigma_{s_n,a_n}\Bigg), \tag{18}$$

where θ_0 is set arbitrarily. In (18), the action a_n is chosen in state s_n according to an ϵ-greedy policy, i.e., with a probability of $(1-\epsilon)$, a greedy action given by $a_n = \text{argmin}_{v\in A(s_n)}\theta_n^T\sigma_{s_n,v}$ is chosen and with probability ϵ, an action in $A(s_n)$ is randomly chosen. Using ϵ-greedy policy for the regular Q-learning algorithm has been well recognized and recommended in the literature (cf. [5, 31]).

5.6 Two-timescale Q-learning based sleep–wake algorithm (TQSA-A)

Although Q-learning with function approximation has been shown to work well in several applications in practice, establishing a proof of convergence of this algorithm is theoretically difficult. A simple counterexample that illustrates the chattering phenomenon when Q-learning is combined with function approximation is provided in [5, Section 6.4]. Moreover, there have also been practical instances where the iterates of QSA-A have been shown to be unstable (cf. [25]).

The problem is complicated due to the *off-policy* nature of QSA-A. The *off-policy* problem here arises because of the presence of the *min* operation in the Q-learning algorithm that introduces nonlinearity in the update rule (see term (I) in (18)). There is also a minor problem of estimating the average cost that involves a min operation as well (see term (II) in (18)). The latter problem can be solved by estimating the average cost in a separate recursion (as we do in (21)) and using this estimate in place of the term (II).

A nested two-loop procedure to overcome the off-policy problem works as follows:

Inner loop Instead of the first min operation (term (I) in (18)), employ a stochastic gradient technique to find the best action that minimizes the approximate Q-value function. A popular scheme for estimating the gradient of a function from simulation is SPSA and we employ a one-simulation SPSA scheme with deterministic perturbations for estimating $\nabla_w Q(s,a)$.

Outer loop instead of the *min* operation, actions are selected according to a given policy, then the Q-learning update would resemble a temporal difference (TD) learning update for the joint (state-action) Markov chain. It has been shown in [32] that TD with linear function approximation converges.

For ensuring convergence of the above procedure, one would have to run the two loops in a serial fashion for sufficiently long duration. This may be time-consuming and also result in slow convergence. To overcome this problem, we employ multi-timescale stochastic approximation [10, Chapter 6] to mimic the two-loop behavior, albeit with different step-sizes for the inner and outer loops. In other words, both the loops are allowed to run simultaneously, with a larger step-size for the inner loop and a smaller one for the outer loop. This achieves the effect of the nested loop procedure, while ensuring rapid convergence.

Recall that we consider a class of parameterized policies satisfying Assumption 1.[4] As illustrated in Fig. 3, the idea in the gradient estimate is to simulate the system with the perturbed policy parameter $w + \delta\Delta$, where $\delta > 0$ is a fixed small constant and $\Delta = (\Delta_1, \ldots, \Delta_N)^T$ are perturbations constructed using certain Hadamard matrices (see Lemma 3.3 of [7] for details of the construction). Given the output

[4] One may use an ϵ-greedy policy for TQSA-A as well, however, that will result in additional exploration. Since TQSA-A updates the parameters of an underlying parameterized Boltzmann policy (which by itself is randomized in nature), we do not need an extra exploration step in our algorithm.

from the perturbed simulation, the gradient of the approximate Q-value function $Q(s,a) \approx \theta^T \sigma_{s,a}$ is estimated as:

$$\nabla_w Q(s,a) \approx \frac{\theta^T \sigma_{s,a}}{\delta} \Delta^{-1}, \tag{19}$$

where $\Delta^{-1} \triangleq (\Delta_1^{-1}, \ldots, \Delta_N^{-1})^T$. It has been shown in [7] that an incremental stochastic recursive algorithm that incorporates the RHS of (19) as its update direction essentially performs a search in the gradient direction when δ is small.

The overall update of the TQSA-A proceeds on two different timescales as follows:

(i) On the faster timescale, the policy parameter is updated along a gradient descent direction using an SPSA estimate (19);

(ii) On the slower timescale, the average cost (4) is estimated and

(iii) Also, on the slower timescale, the Q-value parameter is updated in an on-policy TD algorithm-like fashion.

The update rule for the TQSA-A algorithm is given as follows: $\forall n \geq 0$,

$$\theta_{n+1} = \Gamma_1 \Big(\theta_n + b(n) \sigma_{s_n,a_n} (g(s_n,a_n) - \hat{J}_{n+1} + \theta_n^T \sigma_{s_{n+1},a_{n+1}} - \theta_n^T \sigma_{s_n,a_n}) \Big), \tag{20}$$

$$\hat{J}_{n+1} = \hat{J}_n + c(n) \big(g(s_n,a_n) - \hat{J}_n \big), \tag{21}$$

$$w_{n+1} = \Gamma_2 \left(w_n - a(n) \frac{\theta_n^T \sigma_{s_n,a_n}}{\delta} \Delta_n^{-1} \right). \tag{22}$$

In the above, the choice of features σ_{s_n,a_n} is the same as in the algorithm, QSA-A and is described in Sect. 5.4. $\Gamma_1 : \mathbb{R}^d \to D$, $\Gamma_2 : \mathbb{R}^N \to C$ are certain projection operators that project the iterates θ_n and $w_n, n \geq 1$ to certain prescribed compact and convex subsets D and C of $\mathbb{R}^d$ and $\mathbb{R}^N$, respectively. The recursions (20) and (22) remain stable because of these projection operators, a crucial requirement for convergence of TQSA-A. The step-sizes $b(n), c(n), a(n)$ satisfy the following assumption:

Assumption 2

$$\sum_n a(n) = \sum_n b(n) = \infty, \sum_n (a^2(n) + b^2(n)) < \infty,$$
$$\lim_{n \to \infty} \frac{b(n)}{a(n)} = 0.$$

Further, $c(n) = ka(n)$ for some $k > 0$.

While the first two conditions above are standard in stochastic approximation for step-sizes, the last condition, i.e., $\frac{b(n)}{a(n)} \to 0$ ensures the necessary timescale separation between policy and Q-value parameter updates. In particular, it guarantees that the policy parameter w is updated on the faster timescale and average cost $\hat{J}$ and Q-value parameter θ are updated on the slower timescale.

It turns out that because of the timescale difference, the recursion (22) converges almost surely to a set $w(\theta)$ that is a function of parameter θ and is seen to be a compact subset of $\mathbb{R}^N$. Further, the slower recursion (20) can be seen to track a differential inclusion and converges almost surely to a closed connected internally chain transitive invariant set of this differential inclusion. This claim is made precise by the convergence result in the following section.

5.7 Convergence of TQSA-A

We outline the proof of convergence of the TQSA-A algorithm, with the details being available in an appendix to this paper. In addition to assumptions 1 and 2, we make the following assumption for the analysis.

Assumption 3 The Markov chain induced by any policy w is irreducible and aperiodic.

The above ensures that each state gets visited an infinite number of times over an infinite time horizon and is standard in policy gradient RL algorithms.

The ODE approach is adopted for analyzing the convergence of θ and w recursions (20). In essence, the two-timescale stochastic approximation architecture employed in the TQSA-A algorithm allows (i) the faster timescale analysis of the w-recursion in (20) assuming that the slower θ-recursion is constant (quasi-static), and (ii) the slower timescale analysis of the θ-recursion in (20) assuming that the faster w-recursion has converged. The convergence analysis comprises of the following important steps:

- Theorem 1, in effect, states that the w-recursion performs a gradient descent using one-simulation SPSA and converges to a set of points in the neighborhood of the local minimum of the approximate Q-value function $R(\theta, w)$ (defined below). Note that this analysis is for the w-recursion on the faster timescale, assuming the Q-value function parameter θ to be a constant.
- Analyzing the θ-recursion on the slower timescale, Theorem 2 claims that the iterate θ asymptotically converges to a closed connected internally chain transitive set associated with a corresponding differential inclusion (DI).

We present below the precise statements of these results. Let $\mathcal{C}(C)(\mathcal{C}(D))$ denote the space of all continuous functions from C to $\mathcal{R}^N$ (D to $\mathcal{R}^d$). We define the operator $\hat{\Gamma}_2 : \mathcal{C}(C) \to \mathcal{C}(\mathcal{R}^N)$ as follows:

$$\hat{\Gamma}_2(v(w)) = \lim_{\alpha \downarrow 0} \left(\frac{\Gamma_2(w + \alpha v(w)) - w}{\alpha} \right).$$

Consider the ODE associated with the w-recursion on the faster timescale, assuming $\theta(t) \equiv \theta$ (a constant independent of t):

$$\dot{w}(t) = \hat{\Gamma}_2(-\nabla_w R(\theta, w(t))). \tag{23}$$

Theorem 1 establishes that the w-recursion tracks the above ODE. In the above,

$$R(\theta, w) \stackrel{\Delta}{=} \sum_{i \in S, a \in A(i)} f_w(i,a) \theta^T \sigma_{i,a},$$

where $f_w(i,a)$ are the stationary probabilities $f_w(i,a) = d^{\pi_w}(i)\pi_w(i,a)$, $i \in S$, $a \in A(i)$ for the joint process $\{(X_n, Z_n)\}$, obtained from the state-action tuples at each instant. Here $d^{\pi_w}(i)$ is the stationary probability for the Markov chain $\{X_n\}$ under policy π_w being in state $i \in S$. Let K_θ denote the set of asymptotically stable equilibria of (23), i.e., the local minima of the function $R(\theta, \cdot)$ within the constraint set C. Given $\epsilon > 0$, let K_θ^ϵ denote the ϵ-neighborhood of K_θ, i.e.,

$$K_\theta^\epsilon = \{w \in C \mid \| w - w_0 \| < \epsilon, w_0 \in K_\theta\}.$$

Theorem 1 Let $\theta_n \equiv \theta, \forall n$, for some $\theta \in D \subset \mathcal{R}^d$. Then, given $\epsilon > 0$, there exists $\delta_0 > 0$ such that for all $\delta \in (0, \delta_0]$, $\{w_n\}$ governed by (20) converges the set K_θ^ϵ a.s.

We now analyze the θ-recursion, which is the slower recursion in (20). Let

$T_w : \mathcal{R}^{|S \times A(S)|} \to \mathcal{R}^{|S \times A(S)|}$ be the operator given by

$$T_w(J)(i,a) = g(i,a) - J(\pi_w)e + \sum_{j \in S, b \in A(j)} p_w(i,a;j,b) J(j,b), \tag{24}$$

or in more compact notation

$$T_w(J) = G - J(\pi_w)e + P_w J,$$

where G is the column vector with components $g(i,a), i \in S, a \in A(i)$, $J(\pi_w)$ is the average cost corresponding to the policy parameter w and P_w is the transition probability matrix of the joint (state-action) Markov chain under policy π_w, with components $p_w(i,a;j,b)$. Here $p_w(i,a;j,b)$ denote the transition probabilities of the joint process $\{(X_n, Z_n)\}$. The differential inclusion associated with the θ-recursion of (20) corresponds to

$$\dot{\theta}(t) \in \hat{\Gamma}_\theta(h(\theta)), \tag{25}$$

where $h(\theta)$ is the set-valued map, defined in compact notation as follows:

$$h(\theta) \stackrel{\Delta}{=} \{\Phi^T \mathbb{F}_{w(\theta)}(T_{w(\theta)}(\Phi\theta) - \Phi\theta \mid w \in K_\theta^\epsilon\}.$$

In the above, $\mathbb{F}_w$ denotes the diagonal matrix with elements along the diagonal being $f_w(i,a)$, $i \in S$, $a \in A(i)$. Also, Φ denotes the matrix with rows $\sigma_{s,a}^T, s \in S, a \in A(s)$. The number of rows of this matrix is thus $|S \times A(S)|$, while the number of columns is N. Thus, $\Phi = (\Phi(i), i = 1, \ldots, N)$ where $\Phi(i)$ is the column vector

$$\Phi(i) = (\sigma_{s,a}(i), s \in S, a \in A(s))^T, \quad i = 1, \ldots, N.$$

Further, the projection operator $\hat{\Gamma}_\theta$ is defined as

$$\hat{\Gamma}_\theta \stackrel{\Delta}{=} \cap_{\epsilon > 0} ch\big(\cup_{\|\beta - \theta\| < \epsilon} \{\gamma_1(\beta; y + Y) \mid y \in h(\beta), Y \in R(\beta)\}\big), \text{ where}$$

- $ch(S)$ denotes the closed convex hull of the set S;
- $\gamma_1(\theta; y)$ denotes the directional derivative of Γ_1 at θ in the direction y and is defined by

$$\gamma_1(\theta; y) \stackrel{\Delta}{=} \lim_{\eta \downarrow 0} \left(\frac{\Gamma_1(\theta_n + \eta y) - \theta}{\eta} \right);$$

- $Y(n+1)$ is defined as follows:

$$\begin{aligned} Y(n+1) \stackrel{\Delta}{=} & (g(X_n, Z_n) - J(\pi_{w_n}) \\ & + \theta_n^T \sigma_{X_{n+1}, Z_{n+1}} - \theta_n^T \sigma_{X_n, Z_n}) \sigma_{X_n, Z_n} \\ & - E\big[\big(g(X_n, Z_n) - J(\pi_{w_n}) + \theta_n^T \sigma_{X_{n+1}, Z_{n+1}} \\ & - \theta_n^T \sigma_{X_n, Z_n}) \sigma_{X_n, Z_n} \mid \mathcal{G}(n)\big], \end{aligned}$$

  where $\mathcal{G}(n) = \sigma(\theta_r, X_r, Z_r, r \le n), n \ge 0$ is a sequence of associated sigma fields; and
- $R(\beta)$ denotes the compact support of the conditional distribution of $Y(n+1)$ given $\mathcal{G}(n)$.

The main result is then given as follows:

Theorem 2 The iterate θ_n, $n \ge 0$ governed by (20), converges a.s to a closed connected internally chain transitive invariant set of (25).

The detailed proofs of Theorems 1 and 2 are provided in the supplementary material.

5.8 Intruder's mobility model estimation

The algorithms described in the previous sections assume knowledge of the transition dynamics (the matrix P) of the Markov chain governing the intruder movement. However, in practice, this information is not available. In this section, we present a procedure to estimate P and combine the same with the sleep-wake scheduling algorithms described in the previous section. We assume that P is stationary, i.e., it does not change with time.

The estimation procedure for P is online and convergent. The combination with the sleep-wake scheduling algorithms happens via multi-timescale stochastic approximation. In essence,

we run the estimation procedure for P on the faster timescale while the updates for the parameters of the sleep-wake scheduling algorithms are conducted on the slower timescale. Thus, the update recursions for the individual sleep-wake algorithms see the estimate for P as equilibrated, i.e., converged.

Let $\hat{P}_0$ be the initial estimate of the transition probability matrix P. Then, the estimate $\hat{P}_n$ at time instant n is tuned as follows:

$$\hat{P}_{n+1} = \Pi\left(\hat{P}_n + d(n)\hat{p}_n\hat{p}_{n+1}^T\right). \tag{26}$$

In the above, $\hat{p}_n = [p_n(i) : i = 1, 2, \ldots, N+1]^T$ is a column vector signifying current location of the intruder. Further, $\Pi(\cdot)$ is a projection operator that ensures that the iterates $\hat{P}_n$ satisfy the properties of a transition probability matrix. Also, $\{d(n)\}$ is a step-size sequence chosen such that it is on the faster timescale, while the θ-recursion of the algorithm described earlier is on the slower timescale.

The idea behind the above update rule can be explained as follows: Suppose the locations of the intruder at instants n and $n+1$ are known. Then, $\hat{p}_n$ and $\hat{p}_{n+1}$ would be vectors with the value 1 in l_kth position and 0 elsewhere. The quantity $\hat{p}_n\hat{p}_{n+1}^T$ would thus result in a matrix with 1 at row index l_k and column index l_{k+1} and 0 elsewhere. The recursion (26) then results in a sample averaging behavior (due to stochastic approximation) for estimating the transition dynamics P. The same logic can be extended to the remaining cases, for instance, known l_k and unknown l_{k+1} and so on.

Empirically we observe that the update (26) converges to the true transition probability matrix P for each of the proposed algorithms, in all the network settings considered.

6 Discounted cost setting

We now describe the discounted cost objective. As in the case of the average cost setting (Sect. 4), we describe below the Bellman equation for continuous state-action spaces. However, for the sake of implementation (in later sections), we again use the discrete version of the problem.

For a policy π, define the value function $V^\pi : S \to \mathbb{R}$ as follows:

$$V^\pi(x) = E\left[\sum_{m=0}^{\infty} \gamma^m g(s_m, a_m) \mid X_0 = x\right], \tag{27}$$

for all $x \in \mathcal{S}$. In the above, $\gamma \in (0, 1)$ is a given discount factor. The aim then is to find an optimal value function $V^* : \mathcal{S} \to \mathbb{R}$, i.e.,

$$V^*(x) = \min_{\pi \in \Pi} V^\pi(x) \triangleq V^{\pi^*}(x), \tag{28}$$

where π^* is the optimal policy, i.e., the one for which V^* is the value function. It is well known, see [28], that the optimal value function $V^*(\cdot)$ satisfies the following Bellman equation of optimality in the discounted cost case:

$$V^*(x) = \min_{a \in \mathcal{A}(x)} \left(g(x, a) + \gamma \int p(x, a, dy) V^*(y)\right), \tag{29}$$

for all $x \in \mathcal{S}$. As for the average cost, our algorithms in the discounted cost setting do not require knowledge of the system model and incorporate function approximation.

7 Discounted cost algorithms

In this section, we present two algorithms for sleep-wake scheduling with the goal of minimizing a discounted cost objective described in Sect. 6. The overall structure of both the algorithms follow the schema provided in Algorithm 1. However, in comparison to the average cost algorithms described earlier, the parameter θ is updated in a different fashion here to cater to the discounted cost objective.

7.1 Q-learning based sleep–wake scheduling algorithm (QSA-D)

As in the case of the average cost setting, the Q-learning algorithm cannot be used without employing function approximation because of the size of the state-action space. The function approximation variant of Q-learning in the discounted cost setting parameterizes the Q-values in a similar manner as the average cost setting, i.e., according to (13). The algorithm works with a single online simulation trajectory of states and actions, and updates θ according to

$$\theta_{n+1} = \theta_n + a(n)\sigma_{s_n,a_n}\left(g(s_n, a_n) + \gamma \min_{b \in A(s_{n+1})} \theta_n^T \sigma_{s_{n+1},b} - \theta_n^T \sigma_{s_n,a_n}\right), \tag{30}$$

where θ_0 is set arbitrarily. In the above, s_n and s_{n+1} denote the state at instants n and $n+1$, respectively, and θ_n denotes the n^{th} update of the parameter. In (30), the action a_n is chosen in state s_n according to an $\epsilon-$greedy policy, as in the case of the QSA-A algorithm.

7.2 Two-timescale Q-learning based sleep–wake scheduling algorithm (TQSA-D)

As with the average cost setting, the Q-learning algorithm with function approximation in the discounted setting is not guaranteed to converge because of the off-policy problem. A variant of Q-learning [6] has been recently proposed and has been shown to be convergent. This algorithm uses two-timescale simultaneous perturbation stochastic approximation (SPSA) with Hadamard matrix based deterministic perturbation sequences [7].

The TQSA-D algorithm is a two timescale stochastic approximation algorithm that employs a linear approximation architecture and parameterizes the policy. As in the case of TQSA-A, we assume here that the policy $\pi(s,a)$ is continuously differentiable in the parameter θ, for any state–action pair (s,a). The function approximation parameter θ is tuned on the slower timescale in a TD-like fashion, while the policy parameter w is tuned on the faster timescale in the negative gradient descent direction using SPSA. Let $\pi'_n \stackrel{\triangle}{=} \pi_{(w_n+\delta\Delta_n)} = (\pi_{(w_n+\delta\Delta_n)}(i,a), i \in S, a \in A(i))^T$, where $\delta > 0$ is a given small constant, be the randomized policy parameterized by $(w_n + \delta\Delta_n)$ during the nth instant. Here $\Delta_n, n \geq 0$ are perturbations obtained from the Hadamard matrix based construction described before. The update rule of the TQSA-D algorithm is given as follows: $\forall n \geq 0$,

$$\begin{aligned} \theta_{n+1} &= \Gamma_1\left(\theta_n + b(n)\sigma_{s_n,a_n}\left(r(s_n,a_n) + \gamma\theta_n^T\sigma_{s_{n+1},a_{n+1}} - \theta_n^T\sigma_{s_n,a_n}\right)\right), \\ w_{n+1} &= \Gamma_2\left(w_n - a(n)\frac{\theta_n^T\sigma_{s_n,a_n}}{\delta}\Delta_n^{-1}\right). \end{aligned} \tag{31}$$

The projection operators Γ_1, Γ_2 and the step-sizes $a(n), b(n)$ for all $n \geq 0$ are the same as in TQSA-A and the features σ_{s_n,a_n} are as in the previous algorithms.

8 Simulation setup and results

8.1 Implementation

We implemented our sleep–wake scheduling algorithms - QSA-A and TQSA-A for the average cost setting and QSA-D and TQSA-D for the discounted cost setting, respectively. For the sake of comparison, we also implemented the FCR and Q_{MDP} algorithms proposed in [14]. Note that for each of these algorithms, the knowledge of the mobility model of the intruder is assumed. We briefly recall these algorithms below:

FCR. This algorithm approximates the state evolution (3) by $p_{t+1} = p_t P$, and then attempts to find the sleep time for each sensor by solving the following balance equation:

$$V^{(l)}(p) = \min_u \left(\sum_{j=1}^{u}[pP^j]_l + \sum_{i=1}^{N} c[pP^{u+1}]_i + V^{(l)}(pP^{u+1})\right).$$

Thus, the sleeping policy here is obtained locally for each sensor by solving the above Bellman equation for each sensor, with a strong approximation on the state evolution. Note that our algorithms make no such assumptions and attempt to find the optimal sleeping policy in the global sense (i.e., considering all the sensors) and not in the local sense (i.e., treating the sensors individually).

Q_{MDP}. In this approach, the decomposition into the per sensor problem is the same as in FCR. However here, the underlying assumption is that the location of the object will always be known in the future. Thus, instead of (3), the state evolves here according to $p_{k+1} = e_{l_{k+1}}P$. The objective function for a sensor l, given the state component p, is given by

$$V^{(l)}(p) = \min_u \left(\sum_{j=1}^{u}[pP^j]_l + \sum_{i=1}^{N} c[pP^{u+1}]_i + \sum_{i=1}^{N}[pP^{u+1}]_i V^{(l)}(e_i)\right).$$

The difference between the above and the corresponding equation for FCR is in the third term on the right hand side representing the future cost. In the case of Q_{MDP}, the future cost is the conditional expectation of the cost incurred from the object location after u time units given the current distribution as its location. Thus, one can solve $V^{(l)}(p)$ for any p once $V^{(l)}(e_i), 1 \leq i \leq N$ are known. The Q_{MDP} algorithm then attempts to find a solution using the well-known dynamic programming procedure—policy iteration for MDPs. However, an important drawback with the dynamic programming approaches is the curse of dimensionality (i.e., the computational complexity with solving the associated Markov decision process increases exponentially with the dimension and cardinality of the state and action spaces). RL algorithms that incorporate function approximation techniques alleviate this problem and make the computational complexity manageable, while still ensuring that these algorithms converge to a 'good enough' policy.

8.2 Simulation setup

We perform our experiments on a 2-D network setting (see Fig. 1) of 121 sensors, i.e., a 11×11 grid. The sensor regions overlap here, with each sensor's sensing region overlapping with that of its neighboring nodes. In particular, the sensing regions of sensors in the interior of the grid overlap with eight neighboring nodes.

The simulations were conducted for 6,000 cycles for all algorithms. We set the single-stage cost component c to 0.1 and the discount factor γ to 0.9. For QSA-A/D, we set the exploration parameter ϵ to 0.1. The projection operators $\Gamma_i, i = 1,2$ are chosen such that each co-ordinate of θ and w is forced to evolve within $[1, 100]$. The step-sizes are chosen as follows: For QSA-A, we set $a(n) = \frac{1}{n}, n \geq 1$ and for TQSA-A, we set $b(n) = \frac{1}{n}$, $a(n) = \frac{1}{n^{0.55}}, n \geq 1$, respectively. Further, for TQSA-A/TQSA-D, we set $\delta = 0.001$. For QSA-A, we choose the fixed state s (see (18)) as $\langle p_0, r\rangle$ where p_0 is the initial distribution of p_k and r is a random sleep time vector. It is easy to see that this

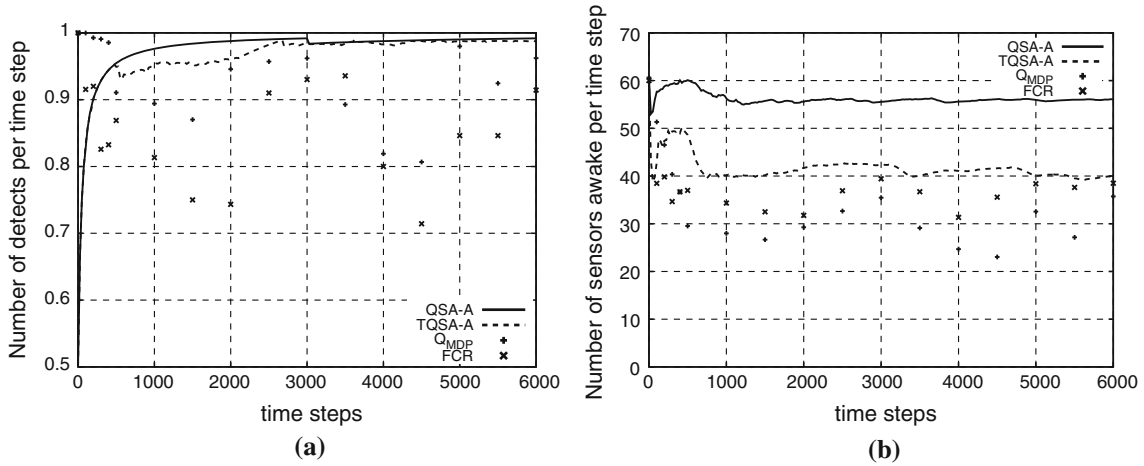

Fig. 4 Tradeoff characteristics—known mobility model (P) case. **a** Number of detects per time step. **b** Number of sensors awake per time step

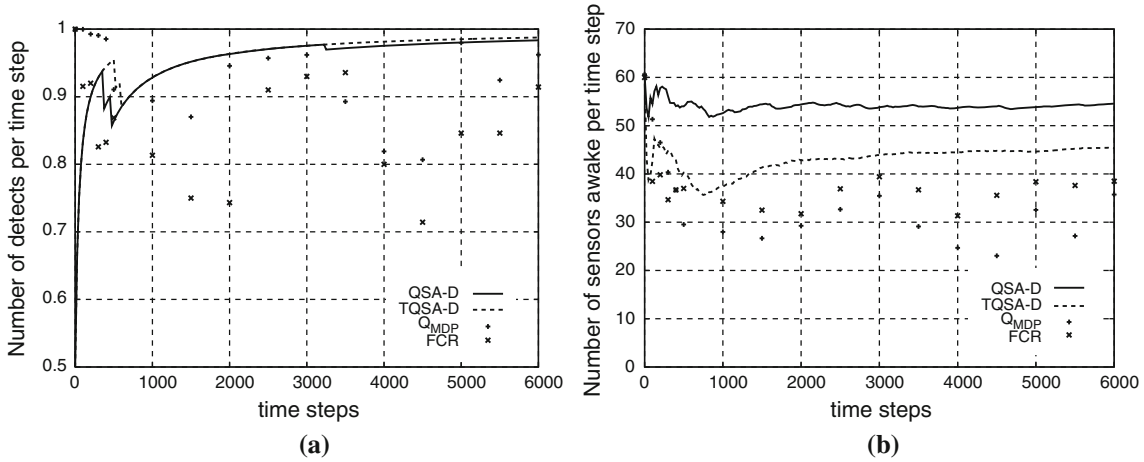

Fig. 5 Tradeoff characteristics in the discounted setting. **a** Number of detects per time step. **b** Number of sensors awake per time step

choice ensures that there is a positive probability of the underlying MDP visiting state s.[5]

8.3 Results

We use the number of sensors awake and the number of detects per time step as the performance metrics for comparing the various sleep/wake algorithms. While the former metric is the ratio of the total number of sensors in the wake state to the number of time-steps, the latter is the ratio of the number of successful detects of the intruder to the number of time-steps. Fig. 4 presents the number of sensors awake and the number of detects per time step, for each of the algorithms studied in the average cost setting, while Fig. 5 presents similar results for the algorithms in the discounted cost setting.

Figure 6a presents the evolution of the Q-value parameter θ for TQSA-A in the average cost setting. Fig. 7 presents the results obtained from the experiments with TQSA-A combined with the mobility model estimation procedure (26). Fig. 6b shows the evolution of the estimate $P_k(i,j)$ of the intruder's mobility model, where i corresponding to the $(6,6)$th cell and j corresponding to $(6,5)$th cell, converges. In contrast, the Q_{MDP} algorithm requires full knowledge of the distribution of the intruder movement and hence, cannot be applied in the setting of unknown P.

8.4 Discussion

We observe that in comparison to the Q_{MDP} algorithm, our algorithms attain a slightly higher tracking accuracy at the cost of a few additional sensors in the wake state. On the

[5] This is because the intruder stays in the starting location for at least one time step and the exploration of actions initially results in a positive probability of a random action being chosen.

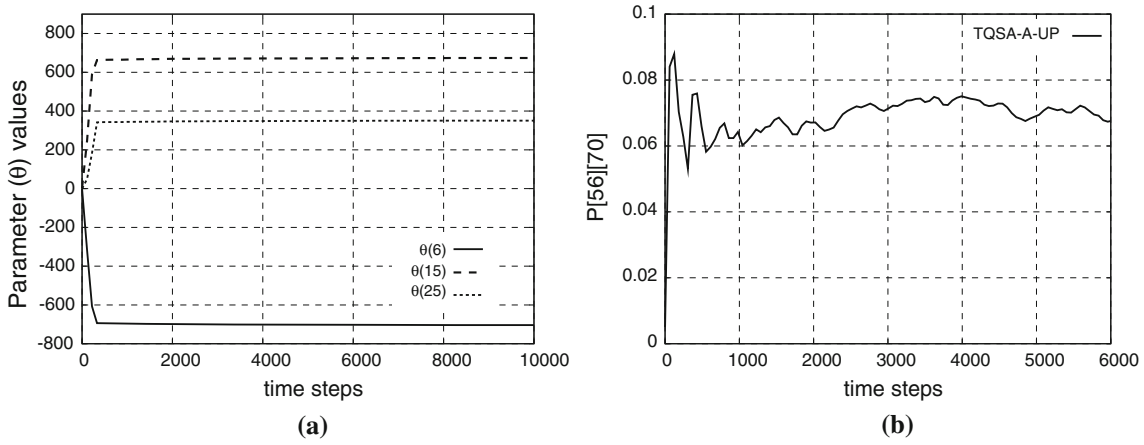

Fig. 6 Convergence trends of θ in TQSA-A with known P and the estimate P_k in TQSA-A with unknown P. **a** Convergence of θ. **b** Convergence of P_k

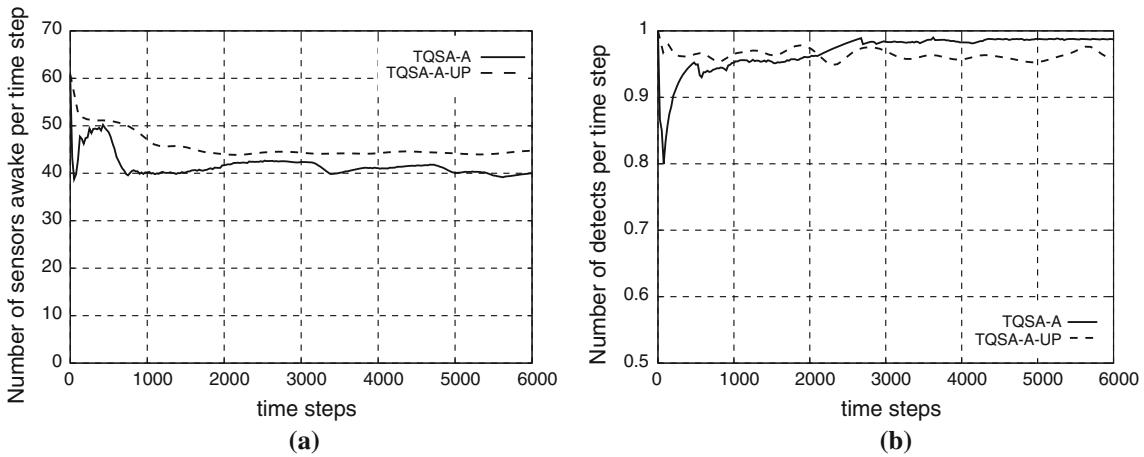

Fig. 7 Tradeoff characteristics for TQSA-A algorithm with known and unknown P. **a** Number of sensors awake per time step. **b** Number of detects per time step

other hand, our algorithms exhibit better tradeoff between energy cost and tracking accuracy in comparison to the FCR algorithm. Amongst our algorithms, we observe that the two timescale variant TQSA-A performs better than the Q-learning based QSA-A, since TQSA-A results in a tracking accuracy similar to QSA-A with lesser number of sensors awake. A similar observation holds in the discounted cost setting as well.

Further, as evident in the tradeoff plot in Fig. 4, the Q_{MDP} algorithm exhibits fluctuating behaviour with a significant number of outliers that show poor tradeoffs. This, we suspect, is due to the underlying requirement of complete future observations in Q_{MDP}. Further, Q_{MDP} (and even FCR) is not a learning algorithm that stabilizes the number of sensors awake and the tracking errors in the long-term. This is because, at each instant, Q_{MDP} attempts to solve the Bellman equation in an approximate fashion and no information about the solution thus obtained is carried forward to the future instants.

On the contrary, our algorithms learn a good enough sleep/wake scheduling policy for the individual sensors with contextual information being carried forward from one time step to the next. This results in a stable regime for the number of sensors awake and the tracking accuracy, unlike Q_{MDP} . While the number of sensors awake for the FCR algorithm is less than that for our algorithms, the tracking accuracy is significantly lower in comparison. For critical tracking systems, where failing to track has higher penalty, our proposed algorithms (esp. TQSA-A) will be able to achieve greater performance (tracking accuracy) at the cost of only a few additional sensors in the wake state.

Further, it is evident from Fig. 6a that the Q-value parameter θ of TQSA-A converges. This is a significant feature of the TQSA-A algorithm as it possesses theoretical convergence

guarantees, unlike QSA-A, which may not converge in some settings. Moreover, it can also be seen that the transient period when the policy parameter θ has not converged, is short. It is worth noting here that providing theoretical rate of convergence results for TQSA-A is difficult. This is because rate results for multi-timescale stochastic approximation algorithms, except for those with linear recursions (see [21]), is not known till date to the best of our knowledge.

We also observe that even for the case when the intruder's mobility model is not known, TQSA-A shows performance on par with the vanilla TQSA-A, which assumes knowledge of P. We also observe that in the TQSA-A algorithm, the estimate P_k of the transition probability matrix P converges to the true P and this is illustrated by the convergence plots in Fig. 6b.

9 Conclusions and future work

We studied the problem of optimizing sleep times in a sensor network for intrusion detection. Following a POMDP formulation similar to the one in [15], our aim in this paper was to minimize certain long-run average and discounted cost objectives. This in turn allowed us to study both transient as well as steady state system behavior. For both the settings considered, we proposed a novel two-timescale Q-learning algorithm with theoretical convergence guarantees. For the sake of comparison, we also developed sleep-scheduling algorithms that are function approximation analogues of the well-known Q-learning algorithm. Next, we extended these algorithms to a setting where the intruder's mobility model is not known. Empirically, we demonstrated the usefulness of our algorithms on a simple two-dimensional network setting.

As future work, one could extend these algorithms to settings where multiple intruders have to be detected. This would involve the conflicting objectives of keeping less number of sensors awake and at the same time, detecting as many intruders as possible. Another interesting direction of future research is to develop intruder detection algorithms in a decentralized setting, i.e., a setting where the individual sensors collaborate in the absence of a central controller. Decentralized variants of our two-timescale Q-learning algorithm TQSA-A can be developed in the following manner: Each sensor runs TQSA-A to decide on the sleep times in a manner similar to the algorithms we propose. However, this would require the knowledge of p_k (distribution of the intruder's location) at each sensor and this can be obtained by means of a message passing scheme between the individual sensors. Since exchanging messages between every pair of sensors may increase the load on the network, a practical alternative is to form (possibly dynamic) groups of sensors, within which the message regarding the intruder's location (or p_k) is exchanged. The individual sensors then decide on the sleep times using this local information and an update rule similar to (20).

References

1. Abounadi, J., Bertsekas, D., & Borkar, V. (2002). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, *40*(3), 681–698.
2. Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In: *ICML*, pp 30–37.
3. Beccuti, M., Codetta-Raiteri, D., & Franceschinis, G. (2009). Multiple abstraction levels in performance analysis of wsn monitoring systems. In: *International ICST conference on performance evaluation methodologies and tools*, p. 73.
4. Bertsekas, D. P. (2007). *Dynamic programming and optimal control* (3rd ed., Vol. II). Belmont: Athena Scientific.
5. Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont: Athena Scientific.
6. Bhatnagar, S., & Lakshmanan, K. (2012). *A new Q-learning algorithm with linear function approximation*. Technical report SSL, IISc, URL http://stochastic.csa.iisc.ernet.in/www/research/files/IISc-CSA-SSL-TR-2012-3.pdf.
7. Bhatnagar, S., Fu, M., Marcus, S., & Wang, I. (2003). Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *13*(2), 180–209.
8. Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2009). Natural actor-critic algorithms. *Automatica*, *45*(11), 2471–2482.
9. Bhatnagar, S., Prasad, H., & Prashanth, L. (2013). *Stochastic recursive algorithms for optimization* (Vol. 434). New York: Springer.
10. Borkar, V. (2008). *Stochastic approximation: A dynamical systems viewpoint*. Cambridge: Cambridge University Press.
11. Cui, Y., Lau, V. K., Wang, R., Huang, H., & Zhang, S. (2012a). A survey on delay-aware resource control for wireless systems-Large deviation theory, stochastic lyapunov drift, and distributed stochastic learning. *IEEE Transactions on Information Theory*, *58*(3), 1677–1701.
12. Cui, Y., Lau, V. K., & Wu, Y. (2012b). Delay-aware BS discontinuous transmission control and user scheduling for energy harvesting downlink coordinated MIMO systems. *IEEE Transactions on Signal Processing*, *60*(7), 3786–3795.
13. Fu, F., & van der Schaar, M. (2009). Learning to compete for resources in wireless stochastic games. *IEEE Transactions on Vehicular Technology*, *58*(4), 1904–1919.
14. Fuemmeler, J., & Veeravalli, V. (2008). Smart sleeping policies for energy efficient tracking in sensor networks. *IEEE Transactions on Signal Processing*, *56*(5), 2091–2101.
15. Fuemmeler, J., Atia, G., & Veeravalli, V. (2011). Sleep control for tracking in sensor networks. *IEEE Transactions on Signal Processing*, *59*(9), 4354–4366.
16. Gui, C., & Mohapatra, P. (2004). Power conservation and quality of surveillance in target tracking sensor networks. In: *Proceedings of the international conference on mobile computing and networking*, pp. 129–143.
17. Jiang, B., Han, K., Ravindran, B., & Cho, H. (2008). Energy efficient sleep scheduling based on moving directions in target tracking sensor network. In: *IEEE international symposium on parallel and distributed processing*, pp. 1–10.
18. Jianlin, M., Fenghong, X., & Hua, L. (2009). RL-based superframe order adaptation algorithm for IEEE 802.15.4 networks. In: *Chinese control and decision conference, IEEE*, pp. 4708–4711.

19. Jin Gy, Lu, & Xy, Park M. S. (2006). Dynamic clustering for object tracking in wireless sensor networks. *Ubiquitous Computing Systems*, *4239*, 200–209.
20. Khan, M. I., & Rinner, B. (2012). Resource coordination in wireless sensor networks by cooperative reinforcement learning. In: *IEEE international conference on pervasive computing and communications workshop*, pp. 895–900.
21. Konda, V. R., & Tsitsiklis, J. N. (2004) Convergence rate of linear two-time-scale stochastic approximation. *Annals of applied probability*, pp. 796–819.
22. Liu, Z., & Elhanany, I. (2006). RL-MAC: A QoS-aware reinforcement learning based MAC protocol for wireless sensor networks. *IEEE International Conference on Networking* (pp. 768–773). IEEE: Sensing and Control.
23. Niu, J. (2010) Self-learning scheduling approach for wireless sensor network. In: *International conference on future computer and communication (ICFCC), IEEE*, Vol. 3, pp. 253–257.
24. Prashanth, L., Chatterjee, A., & Bhatnagar, S. (2014). Adaptive sleep-wake control using reinforcement learning in sensor networks. In: *6th international conference on communication systems and networks (COMSNETS), IEEE*.
25. Prashanth, L. A., & Bhatnagar, S. (2011a). Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In: *14th International IEEE conference on intelligent transportation systems (ITSC)*, pp. 1640–1645.
26. Prashanth, L. A., & Bhatnagar, S. (2011b). Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 412–421.
27. Premkumar, K., & Kumar, A. (2008). *Optimal sleep-wake scheduling for quickest intrusion detection using sensor networks*. Arizona, USA: IEEE INFOCOM.
28. Puterman, M. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.
29. Rucco, L., Bonarini, A., Brandolese, C., & Fornaciari, W. (2013). A bird's eye view on reinforcement learning approaches for power management in WSNs. In: *Wireless and mobile networking conference (WMNC), IEEE*, pp. 1–8.
30. Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, *37*(3), 332–341.
31. Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge: Cambridge University Press.
32. Tsitsiklis, J. N., & Van Roy, B. (1997). An Analysis of Temporal Difference Learning with Function Approximation. *IEEE Transactions on Automatic Control*, *42*(5), 674–690.
33. Watkins, C., & Dayan, P. (1992). Machine learning. *Q-learning*, *8*(3), 279–292.

L. A. Prashanth his Bachelors in Computer Engineering from National Institute of Technology, Surathkal, India, in 2002. He received his Masters and Ph.D. degrees in Computer Science and Automation from Indian Institute of Science, in 2008 and 2013, respectively. From 2002 to 2009, he was with Texas Instruments (India) Pvt Ltd, Bangalore, India, as a Senior Software Systems Engineer working on embedded operating systems and wireless software. Since 2012, he is a Postdoctoral Researcher at INRIA Lille—Team SequeL, hosted by Remi Munos. He was a recipient of the IBM Ph.D. fellowship in 2012. He is the coauthor of a book entitled 'Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods', published by Springer in 2013. His research interests are in stochastic control and optimization, reinforcement learning and multi-armed bandits with applications in networking and transportation domains.

Abhranil Chatterjee received his Bachelors in Computer Science and Engineering from West Bengal University of Technology, India, in 2010. He received his Masters in Systems Science and Automation from Indian Institute of Science, in 2012. From 2012 till date, he has been with Yahoo Software Development (India) Pvt. Ltd., Bangalore, India, as a Senior Software Engineer working on ranking and relevance in multimedia information retrieval systems and intelligent multimedia recommender systems. His research interests are in reinforcement learning with applications in wireless networks and information retrieval systems.

Shalabh Bhatnagar received a Bachelors in Physics (Hons) from the University of Delhi in 1988. He received his Masters and Ph.D degrees in Electrical Engineering from the Indian Institute of Science, Bangalore in 1992 and 1997, respectively. He was a Research Associate at the Institute for Systems Research, University of Maryland, College Park, during 1997 to 2000 and a Divisional Postdoctoral Fellow at the Free University, Amsterdam, during 2000 to 2001. He is currently working as a Professor at the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore. He has also held visiting positions at the Indian Institute of Technology, Delhi and the University of Alberta, Canada. Dr. Bhatnagar's interests are in simulation optimization, stochastic control and reinforcement learning. He has authored or co-authored more than 115 research articles in various journals and conferences. He is also the coauthor of a book with title 'Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods', published by Springer in 2013. He is a Senior Associate of the International Center for Theoretical Physics (ICTP), Italy, a Fellow of the Indian National Academy of Engineering and a Fellow of the Institution of Electronics and Telecommunication Engineers.