Automatica 113 (2020) 108759

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems☆

Alex S. Leong [a,*], Arunselvan Ramaswamy [a], Daniel E. Quevedo [a], Holger Karl [a], Ling Shi [b]

[a] Faculty of Computer Science, Electrical Engineering and Mathematics, Paderborn University, Paderborn, Germany
[b] Department of Electrical and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 1 February 2019
Received in revised form 30 July 2019
Accepted 13 November 2019
Available online 19 December 2019

ABSTRACT

In many cyber–physical systems, we encounter the problem of remote state estimation of geographically distributed and remote physical processes. This paper studies the scheduling of sensor transmissions to estimate the states of multiple remote, dynamic processes. Information from the different sensors has to be transmitted to a central gateway over a wireless network for monitoring purposes, where typically fewer wireless channels are available than there are processes to be monitored. For effective estimation at the gateway, the sensors need to be scheduled appropriately, i.e., at each time instant one needs to decide which sensors have network access and which ones do not. To address this scheduling problem, we formulate an associated Markov decision process (MDP). This MDP is then solved using a Deep Q-Network, a recent deep reinforcement learning algorithm that is at once scalable and model-free. We compare our scheduling algorithm to popular scheduling algorithms such as round-robin and reduced-waiting-time, among others. Our algorithm is shown to significantly outperform these algorithms for many example scenarios.

1. Introduction

Cyber–physical systems (CPS) are systems built through integration of sensors, communication networks, controllers, dynamic (physical) processes and actuators. They are playing an increasingly important role in modern society, in areas such as energy, transportation, manufacturing, and healthcare. The scale of typical CPS such as smart-grids, vehicular traffic networks and smart factories is large. The realization of these systems faces substantial challenges arising in diverse disciplines, ranging from communications and control to computing (Poovendran et al., 2012). Supporting estimation and control applications over wireless networks has posed considerable challenges for the operation of networks and the design of protocols (Johansson, Pappas, Tabuada & Tomlin, 2014).

Fig. 1 illustrates an example of a networked cyber–physical system for the purposes of remote state estimation. A number of processes are observed by sensors, with the sensors sending information via a shared wireless network (consisting of M wireless channels) to a gateway, that computes state estimates of each of these processes. Such situations could, for instance, occur if a central controller wishes to monitor a number of different processes in an industrial plant. From a networking perspective, one challenge lies in scheduling transmissions from the sensors to the gateway, because of both the volatile nature of wireless channels and the need to carefully schedule transmissions over a shared medium (Molisch, 2011). While such channels provide the opportunity for diversity, they also aggravate the dynamic scheduling problem: which channel should be assigned to which sensor, and when? The problem of scheduling is further exacerbated by estimation and control requirements, which may be at odds with typical communication performance parameters such as waiting times and throughput (Chaskar & Madhow, 2003; Wu, Srikant, & Perkins, 2007).

The sensor scheduling problem wherein a single dynamic process is observed by multiple sensors has been studied in e.g. Hovareshti, Gupta, and Baras (2007), Leong, Dey, and Quevedo (2017), Mo, Garone, and Sinopoli (2014) and Zhao, Zhang, Hu, Abate, and Tomlin (2014). More recently, sensor scheduling problems where multiple processes are observed by different sensors have also been investigated (Han, Wu, Zhang, & Shi, 2017; Wu, Ren, Dey, & Shi, 2018). In the case of single channel systems ($M = 1$), optimal sensor scheduling problems without packet drops have

☆ A. Ramaswamy was supported by the German Research Foundation (DFG) - 315248657. L. Shi was supported by a Hong Kong RGC General Research Fund 16204218. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Bert Tanner under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: alex.leong@upb.de (A.S. Leong), arunr@mail.uni-paderborn.de (A. Ramaswamy), dquevedo@ieee.org (D.E. Quevedo), h.karl@upb.de (H. Karl), eesling@ust.hk (L. Shi).

https://doi.org/10.1016/j.automatica.2019.108759
0005-1098/

been previously studied in Han et al. (2017). For the case $M > 1$ and additionally with packet transmission length constraints, some structural results were derived in Wu et al. (2018), however numerical results were only provided for the $M = 1$ case. The focus of the current paper is on the case $M > 1$, where each wireless channel can also experience packet drops. In particular, we want to provide computationally scalable methods for solving optimal sensor scheduling problems.

For the dynamic scheduling problem, the gateway selects at each discrete time instant a subset (of size M) of the N sensors which communicate the sensor readings to the gateway, to update its estimates. We assume that the gateway has knowledge of the process dynamics observed by each sensor, to allow Kalman filter-type estimation algorithms to be run. The scheduling decision could be informed by knowledge about the quality of the estimates as well as by conjectures about channel state and probability of success of transmitting the readings to the gateway. Knowledge of the channel states or channel statistics is not assumed to be known to the gateway (i.e. scheduling is done in a model-free manner), as such knowledge may be expensive to obtain (requiring e.g. the transmission of pilot signals), and furthermore since channel statistics are often also time-varying (Eisen, Gatsis, Pappas, & Ribeiro, 2018).

As previously mentioned, the scale of a CPS is typically large. For our scheduling problem, this leads to an associated MDP with large state and action spaces. Traditional reinforcement learning based algorithms such as Q-learning cannot be used to solve such MDPs due to Bellman's curse of dimensionality (Bertsekas, 2005). The curse of dimensionality can be overcome by the use of function approximations (Sutton & Barto, 2018). Deep Q-Network (DQN) (Mnih et al., 2013, 2015) is one such algorithm using deep neural networks as function approximators, that has shown tremendous promise in solving large MDPs in a scalable, model-free manner. Deep reinforcement learning techniques have also been recently used to study difficult problems arising in control. The work (Demirel, Ramaswamy, Quevedo, & Karl, 2018) studies a similar problem in controller scheduling, however it does not consider packet drops, and requires extra overhead in the transmission of information from the sensors to the scheduler at every time step. The work of Baumann, Zhu, Martius, and Trimpe (2018) studies event-triggered control problems where the communication and control policies are learnt from scratch using an actor-critic approach.

The paper is organized as follows. The system model is presented in Section 2. The sensor scheduling problem and associated MDP is described in Section 3, together with derivation of a stability condition and discussion of computational issues. The proposed deep reinforcement learning approach to the scheduling problem is given in Section 4. Numerical studies can be found in Section 5.

2. System model

2.1. Sensing model

A diagram of the system model is shown in Fig. 1. We consider N independent, linear, discrete-time processes

$$x_{i,k+1} = A_i x_{i,k} + w_{i,k}, \quad i = 1, \dots, N \tag{1}$$

where $x_{i,k} \in \mathbb{R}^{n_{x_i}}$ is the state of process i at time k, and the process noise $w_{i,k}$ is i.i.d. (in time) Gaussian with zero mean and covariance matrix $W_i \geq 0$.[1] Each process is measured by a sensor as

$$y_{i,k} = C_i x_{i,k} + v_{i,k}, \quad i = 1, \dots, N \tag{2}$$

[1] For a symmetric matrix X, we say that $X > 0$ if it is positive definite, and $X \geq 0$ if it is positive semi-definite.

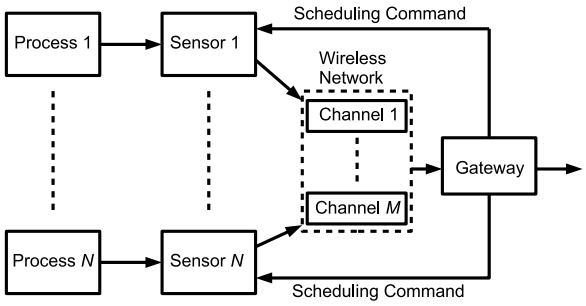

Fig. 1. Remote state estimation with sensor scheduling.

where $y_{i,k} \in \mathbb{R}^{n_{y_i}}$ is the measurement of process i at time k, and the measurement noise $v_{i,k}$ is i.i.d. Gaussian with zero mean and covariance matrix $V_i > 0$. The noise processes $\{w_{i,k}\}$ and $\{v_{j,k}\}$ are assumed to be mutually independent for all i and j.

We assume that each sensor has the computational capability to run a Kalman filter, i.e., each sensor i can compute local state estimates[2] and estimation error covariance matrices

$$\begin{aligned}
\hat{x}^s_{i,k|k-1} &\triangleq \mathbb{E}[x_{i,k} | y_{i,0}, \dots, y_{i,k-1}] \\
\hat{x}^s_{i,k} &\triangleq \mathbb{E}[x_{i,k} | y_{i,0}, \dots, y_{i,k}] \\
P^s_{i,k|k-1} &\triangleq \mathbb{E}[(x_{i,k} - \hat{x}^s_{i,k|k-1})(x_{i,k} - \hat{x}^s_{i,k|k-1})^T | y_{i,0}, \dots, y_{i,k-1}] \\
P^s_{i,k} &\triangleq \mathbb{E}[(x_{i,k} - \hat{x}^s_{i,k})(x_{i,k} - \hat{x}^s_{i,k})^T | y_{i,0}, \dots, y_{i,k}],
\end{aligned}$$

using the Kalman filter equations (Anderson & Moore, 1979). We will assume that every pair (A_i, C_i) is observable, and every pair $(A_i, W_i^{1/2})$ is controllable. Then, the steady-state value of $P^s_{i,k}$ for $k \to \infty$ exists for each sensor, and will be denoted by $\overline{P}_i$. For convenience of presentation, we will assume that the local Kalman filters at the sensors have reached steady state,[3] so that $P^s_{i,k} = \overline{P}_i, \forall i = 1, \dots, N, \forall k$.

2.2. Scheduling and channel model

The sensors wish to transmit their local state estimates $\hat{x}^s_{i,k}$ to a central gateway, which aims to estimate all of the N processes $\{x_{i,k}\}, i = 1, \dots, N$. Sensor transmissions are over a shared wireless network with M channels. In typical applications, $M \ll N$ due to limited resources. Thus, (at most) only M out of the N sensors can transmit at any given time. At each time step k, a scheduler will allocate each of the M channels to one of the sensors. We assume that each channel is allocated to a different sensor, although the case where multiple channels are allocated to the same sensor (e.g. as in Mesquita, Hespanha, and Nair (2012)) can also be handled using our techniques. Define decision variables $a_{m,k} \in \{1, \dots, N\}$ for $m = 1, \dots, M$ as

$$a_{m,k} \triangleq i \text{ if sensor } i \text{ is scheduled to transmit on channel } m \text{ at time } k. \tag{3}$$

Channel transmissions can experience packet drops. Define $\gamma_{m,k} \in \{0, 1\}$ for $m = 1, \dots, M$ such that

$$\gamma_{m,k} \triangleq \begin{cases} 1, & \text{if transmission on channel } m \text{ at time } k \\ & \text{is successfully received at gateway} \\ 0, & \text{otherwise.} \end{cases}$$

[2] In situations where channels experience packet drops, transmission of local state estimates in general gives better estimation performance than transmission of raw measurements (Xu & Hespanha, 2005). It is worth noting that the situation where raw measurements are transmitted can also be handled using the deep Q-learning technique considered in the present work.

[3] Convergence to steady state in general occurs at an exponential rate (Anderson & Moore, 1979).

Each channel is modelled using the Gilbert–Elliott (or Markovian packet drop (Huang & Dey, 2007)) model, with

$$p_m \triangleq \mathbb{P}(\gamma_{m,k} = 0 | \gamma_{m,k-1} = 1),$$
$$q_m \triangleq \mathbb{P}(\gamma_{m,k} = 1 | \gamma_{m,k-1} = 0), \quad m = 1, \dots, M,$$

and with the channels being independent of each other. p_m and q_m are also known, respectively, as the failure rate and recovery rate. As mentioned in the Introduction, we will not assume knowledge of the channel parameters $p_m, q_m, m = 1, \dots, M$ at the scheduler. We note that our model-free approach can also be readily extended to handle more general finite state Markov channels (Quevedo, Østergaard, & Ahlén, 2014; Sadeghi, Kennedy, Rapajic, & Shams, 2008).

2.3. Protocol assumptions

Scheduling is assumed to be done at the gateway, with the decisions $a_{m,k}$ fed back to the sensors.[4] We assume that this (downlink) transmission from gateway to sensor works without errors. We justify this by using all M stochastically independent channels to transmit this signalling information, resulting in an exponentially reduced error probability. Error performance can be further improved by coding across channels (rather than just simple repetition coding) and time (since signalling information is relatively small, time overhead can be invested) (Molisch, 2011; Proakis & Salehi, 2008).

After these channel assignments have been received by the sensors, they send their respective data (local state estimates) to the gateway. Once these (uplink) transmissions are complete, we move to the next time period $k + 1$.

2.4. Remote estimation at gateway

At the gateway, state estimates and estimation error covariances of each of the processes are computed similar to Shi, Epstein, and Murray (2010) and Xu and Hespanha (2005), as follows:

$$\hat{x}_{i,k} = \begin{cases} \hat{x}^s_{i,k}, & \text{if } \exists m \text{ s.t. } a_{m,k} = i \text{ and } \gamma_{m,k} = 1 \\ A_i \hat{x}_{i,k-1}, & \text{otherwise} \end{cases}$$
$$P_{i,k} = \begin{cases} \overline{P}_i, & \text{if } \exists m \text{ s.t. } a_{m,k} = i \text{ and } \gamma_{m,k} = 1 \\ h_i(P_{i,k-1}), & \text{otherwise,} \end{cases} \tag{4}$$

where $h_i(.)$, $i = 1, \dots, N$, is defined as

$$h_i(X) \triangleq A_i X A_i^T + W_i. \tag{5}$$

As mentioned in the Introduction, the gateway is assumed to have knowledge of the parameters for each of the N processes, which allows (4) to be (causally) computed for each process.

3. Problem description

The gateway wishes to find a scheduling policy to minimize the average sum of the trace of the estimation error covariance matrices across all sensors and all times. We will formulate a Markov decision process (MDP) to solve the associated sequential decision making problem:

$$\min_{\{(a_{1,k}, \dots, a_{M,k})\}} \limsup_{T \to \infty} \frac{1}{T} \mathbb{E} \left[\sum_{k=0}^{T-1} \sum_{i=1}^{N} \mathrm{tr} P_{i,k} \right]. \tag{6}$$

We assume that the channel allocations at time k can depend on

$$(P_{1,k-1}, \dots, P_{N,k-1}, \gamma_{1,k-1}, \dots, \gamma_{M,k-1}), \tag{7}$$

namely the estimation error covariances and channel transmission outcomes at the previous time step, which is information that is available to the gateway. From (4) we see that $P_{i,k}$ is always of the form $h_i^n(\overline{P}_i)$ for some $n \in \mathbb{N}$, where $h_i^n(.)$ denotes the n-fold composition of $h_i(.)$ given in (5), with $h_i^0(.)$ being the identity. Define the holding time of sensor i at time k as

$$\tau_{i,k} \triangleq \min\{\tau \geq 0 : \exists m \text{ s.t. } a_{m,k-\tau} = i \text{ and } \gamma_{m,k-\tau} = 1\},$$

which represents the amount of time since the last successful transmission of sensor i to the gateway. Then we can express $P_{i,k}$ as

$$P_{i,k} = h_i^{\tau_{i,k}}(\overline{P}_i),$$

and therefore the channel allocations at time k can, equivalently, depend on

$$(\tau_{1,k-1}, \dots, \tau_{N,k-1}, \gamma_{1,k-1}, \dots, \gamma_{M,k-1}), \tag{8}$$

which is of smaller dimension than (7), as each $\tau_{i,k-1}$ is scalar while each $P_{i,k-1}$ is a matrix. Below we will describe more formally problem (6) as an MDP.

3.1. Formulation as a Markov decision process

State space: From the discussion above, the vector (8) can be regarded as the state[5] of the MDP (6) at time k, and thus the state space is $\mathbb{N}^N \times \{0, 1\}^M$ (where we include 0 in the natural numbers $\mathbb{N}$).

Action space: Next, we have a finite action space

$$\{(a_{1,k}, \dots, a_{M,k}) | a_{1,k}, \dots, a_{M,k} \text{ all distinct}\},$$

corresponding to the $\frac{N!}{(N-M)!}$ different ways of allocating the M channels to the N sensors.

Cost function: Finally, the single stage cost at time k is

$$J_k = \sum_{i=1}^{N} \mathrm{tr} P_{i,k}. \tag{9}$$

Remark 1. As the channel parameters $p_m, q_m, m = 1, \dots, M$ are assumed to be unknown, we do not include the transition probabilities in our formulation of the MDP, and indeed their knowledge is not required when solving the MDP using reinforcement learning methods.

3.2. Stability condition

We will derive a sufficient condition on when the optimal solution to the MDP (6) has bounded average cost, expressed in terms of the process and channel parameters. Such a stability condition is important for reliable monitoring of all of the processes. We first make the following assumption:

Assumption 1. Define $\rho_{\max} \triangleq \max_{i=1,\dots,N} \rho(A_i)$ and $q_{\max} \triangleq \max_{m=1,\dots,M} q_m$, where $\rho(A_i)$ denotes the spectral radius of A_i. We assume that

$$\rho_{\max}^2 (1 - q_{\max}) < 1. \tag{10}$$

[4] Scheduling can also be done inside the network (e.g., at a wireless access point) provided $\gamma_{m,k-1}$ are fed back to the network to allow $P_{i,k-1}$, $i = 1, \dots, N$ to be reconstructed. This makes no difference for the approach considered here.

[5] Note that the state of the MDP is different from the states $x_{i,k}$ of the processes. From now on we will mostly use the word "state" to refer to the state of an MDP.

Theorem 1. *Under Assumption 1, the optimal solution to the MDP (6) has bounded average cost.*

Proof. See the Appendix. ■

Remark 2. For the case of a single process and a single Gilbert–Elliott channel (with transition parameters p and q), when local state estimates are transmitted, a necessary and sufficient condition for bounded expected estimation error covariance is that q satisfies (Gupta, Hassibi, & Murray, 2007):

$$\rho(A)^2(1-q) < 1. \tag{11}$$

The condition (10) can be regarded as a generalization of (11) to multiple processes and multiple channels, and intuitively says that the overall system has bounded cost provided the best channel (in terms of having the largest recovery rate q_m) can keep the expected estimation error covariance of the most unstable process (i.e., having the largest spectral radius) bounded.

3.3. Computational issues

Considering first the case where the channel parameters p_m, $q_m, m = 1, \ldots, M$ are known, numerical solution of (6) using dynamic programming techniques (e.g. using policy iteration or relative value iteration) is in principle possible, after truncating the countable state space $\mathbb{N}^N \times \{0, 1\}^M$ to a finite state space. However in practice, even for relatively small N and M, the sizes of both the state and action spaces can still be considerable, making exact numerical solution infeasible. For the case $M = 1$ without packet drops (and relatively small N in numerical computation), a similar average cost problem has been previously studied (Han et al., 2017). For $M > 1$ and additionally also considering packet transmission length constraints, some structural results were derived in Wu et al. (2018), however numerical results were only provided for the $M = 1$ case.

If the channel parameters $p_m, q_m, m = 1, \ldots, M$, are unknown (and hence the MDP transition probabilities are also unknown), as is assumed in the current work, then standard dynamic programming approaches for solving MDPs cannot be used.

In order to overcome the above mentioned problems of large state space and unknown channel parameters, we will use recently developed reinforcement learning (Q-learning) methods utilizing deep neural networks for function approximation (Mnih et al., 2013, 2015), which will be described in the next section.

4. Sensor scheduling using deep reinforcement learning

Consider the discounted cost problem

$$\min_{\{(a_{1,k},\ldots,a_{M,k})\}} \limsup_{T\to\infty} \mathbb{E}\left[\sum_{k=0}^{T-1}\sum_{i=1}^{N} \delta^k \mathrm{tr} P_{i,k}\right] \tag{12}$$

where $\delta < 1$ is a discount factor. In this paper we will approximate the solution to problem (6) by solving (12) using reinforcement learning techniques, with a discount factor δ close to 1 (Hernández-Lerma & Lasserre, 1996). While Q-learning type algorithms for average reward maximization problems exist (Abounadi, Bertsekas, & Borkar, 2001; Bertsekas, 2012), most reinforcement learning algorithms assume a discounted setting, in particular the deep reinforcement learning techniques of Mnih et al. (2013, 2015). A more formal justification for solving the discounted cost problem will be given in Section 4.2.

4.1. Solving the discounted cost problem using deep reinforcement learning

Let us rewrite (12) as the equivalent discounted reward maximization problem:

$$\max_{\{(a_{1,k},\ldots,a_{M,k})\}} \liminf_{T\to\infty} \mathbb{E}\left[\sum_{k=0}^{T-1}\sum_{i=1}^{N} -\delta^k \mathrm{tr} P_{i,k}\right]. \tag{13}$$

The Q-factor or action–value function $Q(s, a)$ represents the expected future reward associated with taking action a when at state s (Bertsekas, 2012; Sutton & Barto, 2018). The Q-factor version of the Bellman equation for problem (13) is:

$$Q^*(s,a) = \mathbb{E}\left[r + \delta \max_{a'} Q^*(s', a') \middle| s, a\right],$$

where s' represents the value of the next state given the current state s and action a, and $Q^*(.,.)$ are the optimal Q-factors. If we know $Q^*(.,.)$, then we can find a corresponding optimal stationary policy, with action $a^*(s)$ for each state s as follows:

$$a^*(s) = \mathrm{argmax}_a Q^*(s, a).$$

The well-known Q-learning algorithm will, in principle, converge to the optimal Q-factors, but in practice the convergence is rather slow and requires both the state and action spaces to be small in order for the method to be feasible. For large MDPs one can approximate $Q^*(s, a)$ by a function $Q(s, a; \theta)$ parameterized by a set of weights θ (Sutton & Barto, 2018), and then learning these weights. Deep reinforcement learning refers to the case where the function approximation $Q(s, a; \theta)$ uses a (deep) neural network, which has been crucial in recent key breakthroughs in artificial intelligence such as in the playing of Go (Silver et al., 2016). The deep Q-learning techniques introduced in Mnih et al. (2013, 2015) also included a number of important innovations aimed at stabilizing the learning algorithm, in particular (1) the notion of experience replay[6] (see step 9 of Algorithm 1), and (2) fixing the target Q-network at regular intervals[7] (see step 12 of Algorithm 1). Based on these ideas, our approach to solving problem (13) is given as Algorithm 1.

In Algorithm 1,

$$a_t = (a_{1,t}, \ldots, a_{M,t}),$$

c.f. (3), corresponds to the allocation of the M channels at time t, and the single stage reward is given by

$$r_t = \sum_{i=1}^{N} -\mathrm{tr} P_{i,t}.$$

The state s_t could be chosen as

$$s_t = (\tau_{1,t-1}, \ldots, \tau_{N,t-1}, \gamma_{1,t-1}, \ldots, \gamma_{M,t-1})$$

as in Section 3.1, however for the simulations in Section 5 we further augment the state to

$$s_t = \big(\tau_{1,t-1}, \ldots, \tau_{N,t-1}, \mathrm{tr}(h_1(P_{1,t-1})), \ldots, \mathrm{tr}(h_N(P_{N,t-1})), \gamma_{1,t-1}, \ldots, \gamma_{M,t-1}\big), \tag{14}$$

where $\mathrm{tr}(h_i(P_{i,t-1}))$ is directly related to the reward function at time t when we do not receive transmission from sensor i, which we have found in some cases gives faster convergence for the

[6] In experience replay we store the agent's experiences at each time-step, pooled over many episodes, into a replay memory. During the minibatch updates, random samples from the replay memory are drawn. Such a technique can reduce correlations in the observation data.

[7] This technique can reduce correlations between the Q-factors and the target.

Algorithm 1 Deep Q-network for wireless sensor scheduling

1: Initialize replay memory $\mathcal{D}$ to capacity K
2: Initialize network Q with random weights θ_0
3: Initialize target network $\hat{Q}$ with weights $\theta^- = \theta_0$
4: Initialize s_0
5: **for** $t = 0, 1, \ldots, T$ **do**
6: With probability ε select a random action a_t, otherwise select $a_t = \text{argmax}_a Q(s_t, a; \theta_t)$
7: Execute a_t, and observe r_t and s_{t+1}
8: Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
9: Sample random mini-batch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
10: Set $z_j = r_j + \delta \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-)$ for each sample in mini-batch
11: Perform a mini-batch gradient descent step on $(z_j - Q(s_j, a_j; \theta_t))^2$ to obtain θ_{t+1}
12: Every c steps set $\theta^- = \theta_t$
13: **end for**

algorithm. For details of the hyper-parameters for Algorithm 1 used in this paper, see Section 5. We note that Algorithm 1 can be run online, and is model-free in that it does not need knowledge of the channel parameters $p_m, q_m, m = 1, \ldots, M$.

4.2. Relationship to average cost problem

As stated in Section 3, the aim of the scheduler is to find a scheduling policy that minimizes the average estimation error covariances, i.e., solves an associated average cost problem. If the communication channels satisfy Assumption 1, then it follows from Theorem 1 that there exists a scheduling policy that ensures that the cost is bounded. In this subsection, we show that the policy found by solving the associated discounted cost problem is an ϵ-optimal policy for the average cost problem.[8] Furthermore, ϵ can be made arbitrarily small by controlling the discount factor, δ, of the associated MDP.

Recall that J_k given by (9) is the single stage cost associated with problem (6). Before proceeding, we state Abel's theorem (Hernández-Lerma & Lasserre, 1996) for our setting:

Theorem 2 (*Abel*). *Let $\{J_k\}_{k\geq 0}$ be a sequence of positive real numbers. Then*

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{k=0}^{T-1} J_k \leq \liminf_{\delta\uparrow 1}(1-\delta)\sum_{k=0}^{\infty}\delta^k J_k$$
$$\leq \limsup_{\delta\uparrow 1}(1-\delta)\sum_{k=0}^{\infty}\delta^k J_k \leq \limsup_{T\to\infty}\frac{1}{T}\sum_{k=0}^{T-1} J_k.$$

From Theorem 1 it follows that there exist (stabilizing) scheduling policies with finite associated average costs. It now follows from Abel's theorem that:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{k=0}^{T-1} J_k = \lim_{\delta\uparrow 1}(1-\delta)\sum_{k=0}^{\infty}\delta^k J_k < \infty. \tag{15}$$

Furthermore, given $\epsilon > 0$, there exists an $\delta(\epsilon) \approx 1$, dependent on ϵ, such that:

$$\lim_{\delta\uparrow 1}(1-\delta)\sum_{k=0}^{\infty}\delta^k J_k \leq (1-\delta(\epsilon))\sum_{k=0}^{\infty}\delta(\epsilon)^k J_k + \epsilon,$$

$$\Rightarrow \lim_{T\to\infty}\frac{1}{T}\sum_{k=0}^{T-1} J_k \leq (1-\delta(\epsilon))\sum_{k=0}^{\infty}\delta(\epsilon)^k J_k + \epsilon.$$

In addition to ϵ, $\delta(\epsilon)$ also depends on the actual realizations of the single stage cost sequences $\{J_k\}_{k\geq 0}$. If one wishes to find an ϵ-optimal policy, then one can choose a discount factor $\delta(\epsilon)$, provided the "orders" of these single stage costs are known. In our problem, the single stage costs are unbounded. However, it is clear that the discount factor $\delta \uparrow 1$ as $\epsilon \downarrow 0$. Hence, in our numerical experiments, we choose a discount factor close to 1.

5. Numerical studies

We consider an example with $N = 6$ sensors and $M = 3$ channels. Each process has state dimension 2 (i.e. $n_{x_i} = 2, i = 1, \ldots, N$) and scalar measurements ($n_{y_i} = 1, i = 1, \ldots, N$). The process parameters $A_i, C_i, W_i, V_i, i = 1, \ldots, N$ and channel transition probabilities $p_m, q_m, m = 1, \ldots, M$ are randomly generated. The eigenvalues of A_i are drawn uniformly from the range $(0, 1.3)$. The entries of C_i are drawn uniformly from the range $(0, 1)$, and W_i and V_i are generated by random orthogonal transformations of a diagonal matrix with random diagonal entries drawn uniformly from the range $(0.2, 1.0)$. The channel transition probabilities p_m and q_m are uniformly generated from the range $(0, 1)$.

The following hyper-parameters for Algorithm 1 are used in our simulations. In the deep-Q network, the augmented state (14) of dimension $2N + M$ is fed in as input, i.e. there is an input layer with $2N + M = 15$ nodes. We use two hidden layers, with each hidden layer having 1024 nodes, and a fully connected layer with outputs for each of the $N!/(N-M)! = 120$ actions. The discount factor is set to $\delta = 0.95$. The experience replay memory has size $K = 20\,000$. The exploration parameter ε in step 6 of Algorithm 1 is attenuated from 1 to 0.01 at the rate of 0.999, i.e. $\varepsilon \leftarrow \max(0.999\varepsilon, 0.01)$ after every iteration. In the neural network training (step 11 of Algorithm 1) the ADAM optimizer (Kingma & Ba, 2015) is used with an initial learning rate of e^{-4} and a learning rate decay of 0.001.[9] The size of each mini-batch is 32. The target Q-network is updated once every $c = 100$ time steps.

Algorithm 1 is run to train our deep Q-network. In order to get a better idea of the training quality over time, we will reset the process after each $T = 500$, which we will refer to as an episode (Sutton & Barto, 2018). Running on a standard Intel Core i7 4790 with 8 Gb RAM (without GPU), each episode of training when using the above hyper-parameters took around 30 s to complete. The empirical average cost

$$\frac{1}{T}\sum_{k=0}^{T-1}\sum_{i=1}^{N}\text{tr}P_{i,k}$$

over different episodes for one randomly generated set of parameters is plotted in Fig. 2.

We stopped training after 200 episodes. We then use the trained $Q(.,.;\theta)$ to generate a policy according to

$$a^*(s) = \text{argmax}_a Q(s, a; \theta).$$

Using the trained policy, simulating the process over 50 000 time steps then gives an empirical average cost of around 17.8. We compare this performance with the following policies:

(1) A random policy that at each time k randomly allocates M out of the N sensors to the M channels.

[8] Note that ϵ here is different from the exploration parameter ε of Algorithm 1.

[9] If α_t represents the learning rate at iteration t, α_0 the initial learning rate, and d the decay, then $\alpha_t = \frac{\alpha_0}{1+dt}$.

Table 1
Empirical average costs for 10 randomly generated sets of parameters.

Param. Set	Random	Round Robin	Greedy holding time	Greedy error covariance	Deep RL	No replay, no target Q
1	29151	954	55.7	26.2	21.5	22.1
2	1612	415	80.8	49.4	36.4	41.2
3	2358	722	80.4	51.7	32.8	44.3
4	136	82.7	47.4	39.9	34.3	36.7
5	102	42.8	17.1	13.5	10.4	10.6
6	119	34.9	19.3	18.1	15.7	16.8
7	10097	2576	58.4	42.1	35.8	39.5
8	65630	12555	136	77.4	28.7	29.3
9	37.2	30.7	25.9	23.2	21.8	22.5
10	29321	9049	99.4	64.6	36.7	37.7

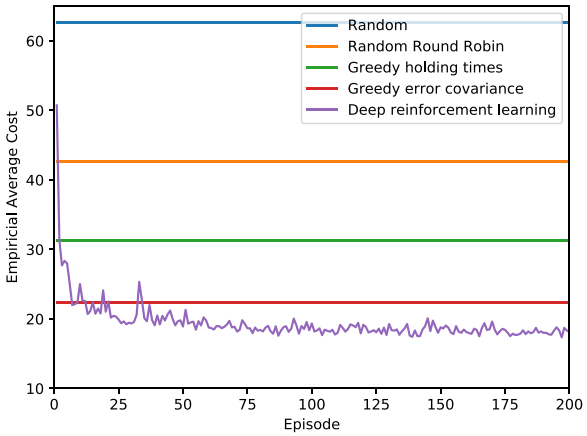

Fig. 2. Empirical average cost over different training episodes. The long term average performances of other suboptimal algorithms are also shown for comparison.

(2) A round robin policy where M successive sensors (modulo N) are randomly allocated to the M channels at every time instance.[10]
(3) A greedy policy on the holding times, where at each time k we allocate the M sensors with the largest $\tau_{i,k-1}$ (in the case of ties we take the sensors with smallest indices) randomly to the M channels.
(4) A greedy policy on the error covariance, where at each time k we allocate the M sensors with the largest $\mathrm{tr}P_{i,k-1}$ randomly to the M channels.

Simulation over 50 000 time steps gives an empirical average cost of around 62.7 for the random policy, 42.7 for the round robin policy, 31.3 for the greedy policy on holding times, and 22.4 for the greedy policy on error covariances. The performances of these policies are also shown in Fig. 2 for comparison. We see that our deep reinforcement learning approach consistently outperforms these policies after around 40–50 episodes of training.

In Table 1 we report further comparisons between the random policy, round robin policy, greedy policies, and the performance using deep reinforcement learning, for 10 different randomly generated sets of parameters $A_i, C_i, W_i, V_i, p_m, q_m, i = 1, \ldots, N, m = 1, \ldots, M$ (making sure that condition (10) is satisfied), while keeping $N = 6$ and $M = 3$. The same hyper-parameters for training the deep Q-network as in the above were used. We can see that the random policy and round robin policy generally do not perform well (although the performance of the round robin policy seems to be better than the purely random policy), and in fact appear to lead to instability in some of the scenarios. The greedy policy on the error covariances performs better than the greedy policy on the holding times, due to the use of more knowledge of the system parameters. We also see that in each scenario the approach using deep reinforcement learning performs significantly better than all the other considered policies. The last column of Table 1 gives the performance when the techniques from Mnih et al. (2013) and Mnih et al. (2015) of experience replay and fixing the target Q-network are not used. We see that without using these techniques, while in some cases the performance is similar, in other cases there is a significant performance loss.

Remark 3. Existing non-control aware scheduling strategies include random, round robin, or greedy strategies with respect to a given parameter, which are also used to, e.g., reduce waiting/holding times. However, in estimation and control applications such strategies do not perform as well as strategies which take into account the dynamics of the processes, as can be seen in Table 1.

6. Conclusion

This paper has studied a sensor scheduling problem for allocating wireless channels to sensors, for the purposes of remote state estimation of multiple dynamical systems. With the aim of providing a method which can handle larger problems than previous work in the literature, we have proposed an approach based on modern deep reinforcement learning ideas. The resulting scheduling algorithm can be run online, and is model-free with respect to the wireless channel parameters. Numerical results have demonstrated that our approach consistently and significantly outperforms other suboptimal sensor scheduling policies. Future work will include the study of model-based reinforcement learning techniques (Pong, Gu, Dalal, & Levine, 2018), to possibly improve the speed of learning when additional knowledge about the channel parameters is available.

Appendix. Proof of Theorem 1

In the case $\rho_{\max} < 1$, condition (10) is always satisfied. Indeed, in this case each process is stable and so the MDP (6) has bounded average cost even when there are no sensor transmissions.

Thus we concentrate on the case $\rho_{\max} \geq 1$. Let

$$m^* \triangleq \arg \max_{m=1,\ldots,M} q_m.$$

First assume a single channel system where only channel m^* is available. Consider a suboptimal policy where at each time

[10] Round robin schedules are similar to periodic schedules commonly studied in the control literature when there are no packet drops (Mo et al., 2014; Zhao et al., 2014).

instant the sensor with the largest holding time is chosen to transmit, provided that this holding time is greater than some $L > 2N$ (Mesquita et al., 2012). Using an argument similar to the proof of the first part of Theorem 3 in Mesquita et al. (2012), we can show that this policy has bounded average cost if

$$\rho_{\max}^2 P_L^{1/L} < 1, \tag{A.1}$$

where P_L can be expressed as

$$P_L = \sum_{n<N} \mathbb{P}(n \text{ successful transmissions in } L \text{ time steps}).$$

The rest of the argument in Theorem 3 of Mesquita et al. (2012) assumes i.i.d. packet dropping channels. To extend the argument to Markovian packet drops as considered in the current work, we make the following observation: Given that there are n successful transmissions, then there will be $L-n$ failed transmissions in these L time steps. Of these $L-n$ failed transmissions, *at most n of them will have followed a successful transmission* (or equivalently at least $L-2n$ of them will have followed a failed transmission). From this observation, we have

$$\begin{aligned} P_L &= \sum_{n<N} \mathbb{P}(n \text{ successful transmissions in } L \text{ time steps}) \\ &\le \sum_{n<N} \binom{L}{n} (\max(q_{m^*}, 1-p_{m^*}))^n \\ &\quad \times (\max(p_{m^*}, 1-q_{m^*}))^n (1-q_{m^*})^{L-2n} \\ &\le (N-1)\binom{L}{N-1}(1-q_{m^*})^{L-2n}. \end{aligned} \tag{A.2}$$

In the first inequality in (A.2), the term $(\max(q_{m^*}, 1-p_{m^*}))^n$ upper bounds the probability of having n successful transmissions, while the term $(\max(p_{m^*}, 1-q_{m^*}))^n(1-q_{m^*})^{L-2n}$ upper bounds the probability of having $L-n$ failed transmissions, with at least $L-2n$ also having the previous transmission fail. The second inequality in (A.2) holds as $\binom{L}{n} \le \binom{L}{N-1}$ for all $n < N$ if $L > 2N$. Taking limits in (A.2) gives

$$\begin{aligned} \lim_{L\to\infty} P_L^{1/L} &\le \lim_{L\to\infty} (N-1)^{1/L} \binom{L}{N-1}^{1/L} (1-q_{m^*})^{(L-2n)/L} \\ &= 1 - q_{m^*}. \end{aligned}$$

Then by Assumption 1, the condition (A.1) can always be satisfied for L sufficiently large, and so the suboptimal policy has bounded average cost. Thus the MDP (6) with only the single channel m^* has bounded optimal average cost. As utilizing additional channels does not increase the optimal average cost, the result follows.

References

Abounadi, J., Bertsekas, D., & Borkar, V. S. (2001). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, *40*(3), 681–698.

Anderson, B. D. O., & Moore, J. B. (1979). *Optimal filtering*. New Jersey: Prentice Hall.

Baumann, D., Zhu, J.-J., Martius, G., & Trimpe, S. (2018). Deep reinforcement learning for event-triggered control. In *Proc. IEEE conf. decision and control*.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control: Vol. I* (3rd ed.). Massachusetts: Athena Scientific.

Bertsekas, D. P. (2012). *Dynamic programming and optimal control: Vol. II* (4th ed.). Massachusetts: Athena Scientific.

Chaskar, H. M., & Madhow, U. (2003). Fair scheduling with tunable latency: a round-robin approach. *IEEE/ACM Transactions on Networking*, *11*(4), 592–601.

Demirel, B., Ramaswamy, A., Quevedo, D. E., & Karl, H. (2018). DeepCAS: A Deep reinforcement learning algorithm for control-aware scheduling. *IEEE Control Systems Letters*, *2*(4), 737–742.

Eisen, M., Gatsis, K., Pappas, G. J., & Ribeiro, A. (2018). Learning in non-stationary wireless control systems via Newton's method. In *Proc. American control conf.*.

Gupta, V., Hassibi, B., & Murray, R. M. (2007). Optimal LQG control across packet-dropping links. *Systems & Control Letters*, *56*, 439–446.

Han, D., Wu, J., Zhang, H., & Shi, L. (2017). Optimal sensor scheduling for multiple linear dynamical systems. *Automatica*, *75*, 260–270.

Hernández-Lerma, O., & Lasserre, J. B. (1996). *Discrete-time Markov control processes: basic optimality criteria*. New York: Springer-Verlag.

Hovareshti, P., Gupta, V., & Baras, J. S. (2007). Sensor scheduling using smart sensors. In *Proc. IEEE conf. decision and control*.

Huang, M., & Dey, S. (2007). Stability of Kalman filtering with Markovian packet losses. *Automatica*, *43*, 598–607.

Johansson, K. H., Pappas, G. J., Tabuada, P., & Tomlin, C. J. (Eds.), (2014). Special issue on control of cyber-physical systems. *IEEE Transactions on Automatic Control*, *59*(12).

Kingma, D. P., & Ba, J. L. (2015). Adam: a method for stochastic optimization. In *Proc. ICLR*.

Leong, A. S., Dey, S., & Quevedo, D. E. (2017). Sensor scheduling in variance based event triggered estimation with packet drops. *IEEE Transactions on Automatic Control*, *62*(4), 1880–1895.

Mesquita, A. R., Hespanha, J. P., & Nair, G. N. (2012). Redundant data transmission in control/estimation over lossy networks. *Automatica*, *48*, 1612–1620.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. In *Proc. NIPS deep learning workshop*.

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Mo, Y., Garone, E., & Sinopoli, B. (2014). On infinite-horizon sensor scheduling. *Systems & Control Letters*, *67*, 65–70.

Molisch, A. F. (2011). *Wireless communications* (2nd ed.). John Wiley & Sons.

Pong, V., Gu, S., Dalal, M., & Levine, S. (2018). Temporal difference models: model-free deep RL for model-based control. In *Proc. ICLR*.

Poovendran, R., Sampigethaya, K., Gupta, S. K. S., Lee, I., Prasad, K. V., Corman, D., & Paunicka, J. L. (Eds.), (2012). Special issue on cyber-physical systems. *Proceedings of IEEE*, *100*(1).

Proakis, J. G., & Salehi, M. (2008). Digital communications, (5th ed.). New York: McGraw-Hill.

Quevedo, D. E., Østergaard, J., & Ahlén, A. (2014). Power control and coding formulation for state estimation with wireless sensors. *IEEE Transactions on Control Systems Technology*, *22*(2), 413–427.

Sadeghi, P., Kennedy, R. A., Rapajic, P. B., & Shams, R. (2008). Finite-state Markov modeling of fading channels. *IEEE Signal Processing Magazine*, *25*(5), 57–80.

Shi, L., Epstein, M., & Murray, R. M. (2010). Kalman filtering over a packet-dropping network: A probabilistic perspective. *IEEE Transactions on Automatic Control*, *55*(3), 594–604.

Silver, D., Huang, A., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning* (2nd ed.). Massachusetts: The MIT Press.

Wu, S., Ren, X., Dey, S., & Shi, L. (2018). Optimal scheduling of multiple sensors over shared channels with packet transmission constraint. *Automatica*, *96*, 22–31.

Wu, X., Srikant, R., & Perkins, J. R. (2007). Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. *IEEE Transactions on Mobile Computing*, *6*(6), 595–605.

Xu, Y., & Hespanha, J. P. (2005). Estimation under uncontrolled and controlled communications in networked control systems. In *Proc. IEEE conf. decision and control* (pp. 842–847).

Zhao, L., Zhang, W., Hu, J., Abate, A., & Tomlin, C. J. (2014). On the optimal solutions of the infinite-horizon linear sensor scheduling problem. *IEEE Transactions on Automatic Control*, *59*(10), 2825–2830.

Alex S. Leong was born in Macau in 1980. He received the B.S. degree in mathematics and B.E. degree in electrical engineering in 2003, and the Ph.D. degree in electrical engineering in 2008, all from the University of Melbourne, Australia.

He is currently a Research Associate at Paderborn University, Germany. He was with the Department of Electrical and Electronic Engineering at the University of Melbourne from 2008 to 2015. His research interests include networked control systems, signal processing for sensor networks, and statistical signal processing.

He was the recipient of the L. R. East Medal from Engineers Australia in 2003, an Australian Postdoctoral Fellowship from the Australian Research Council in 2009, and a Discovery Early Career Researcher Award from the Australian Research Council in 2012.

Arunselvan Ramaswamy is currently a Post-Doctoral Researcher at the Department of Electrical Engineering and Information Technology, Paderborn University, Germany. His position is fully funded by the Special Priority Programme 1914 of the German Research Foundation (DFG).

He completed MSc. (Engg.) in 2012 and Ph.D. in 2017 from the Department of Computer Science and Automation, Indian Institute of Science (IISc), India. His Ph.D. thesis received special commendation for outstanding research from IISc. As a graduate student he interned at IBM-Research India, where he worked on big data optimization. His primary areas of research include stochastic approximation algorithms, dynamical systems, reinforcement learning, deep learning, stochastic optimal control and stochastic processes.

Daniel E. Quevedo is Head of the Chair of Automatic Control (Regelungs- und Automatisierungstechnik) at Paderborn University, Germany. He received Ingeniero Civil Electrónico and M.Sc. degrees from the Universidad Técnica Federico Santa María, Chile, in 2000. In 2005, he was awarded the Ph.D. degree from the University of Newcastle in Australia.

He was supported by a full scholarship from the alumni association during his time at the Universidad Técnica Federico Santa María and received several university-wide prizes upon graduating. He received the IEEE Conference on Decision and Control Best Student Paper Award in 2003 and was also a finalist in 2002. In 2009 he was awarded a five-year Research Fellowship from the Australian Research Council. He is co-recipient of the 2018 IEEE Transactions on Automatic Control George S. Axelby Outstanding Paper Award.

He is Associate Editor of the IEEE Control Systems Magazine, member of the Editorial Board of the International Journal of Robust and Nonlinear Control, and past Chair of the IEEE Control Systems Society Technical Committee on Networks & Communication Systems. His research interests are in control of networked systems and of power converters.

Holger Karl has obtained his Ph.D. in Computer Science from Humboldt University of Berlin in 1999. From 2000 to 2004, he was postdoctoral researcher at Technical University Berlin. In 2004, he joined Paderborn University as professor of computer science, where he leads the Computer Networks research group.

Ling Shi received the B.S. degree in electrical and electronic engineering from Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2002 and the Ph.D. degree in Control and Dynamical Systems from California Institute of Technology, Pasadena, CA, USA, in 2008. He is currently an associate professor at the Department of Electronic and Computer Engineering, and the associate director of the Robotics Institute, both at the Hong Kong University of Science and Technology. His research interests include cyber–physical systems security, networked control systems, sensor scheduling, event-based state estimation and exoskeleton robots. He is a senior member of IEEE. He served as an editorial board member for The European Control Conference 2013–2016. He was a subject editor for International Journal of Robust and Nonlinear Control (2015–2017). He has been serving as an associate editor for IEEE Transactions on Control of Network Systems from July 2016, and an associate editor for IEEE Control Systems Letters from Feb 2017. He also served as an associate editor for a special issue on Secure Control of Cyber–Physical Systems in the IEEE Transactions on Control of Network Systems in 2015–2017. He served as the General Chair of the 23rd International Symposium on Mathematical Theory of Networks and Systems (MTNS 2018).