Deep Reinforcement Learning Sensor Scheduling for Effective Monitoring of Dynamical Systems

Mohammad Alali[a,*], Armita Kazeminajafabadi[a] and Mahdi Imani[a]

[a]Northeastern University, 360 Huntington Ave, Boston, MA, 02115, U.S.

ARTICLE INFO

Keywords:
Sensor Scheduling
Monitoring
Hidden Markov Models
State Estimation
Reinforcement Learning.

ABSTRACT

Advances in technology have enabled the use of sensors with varied modalities to monitor different parts of systems, each providing diverse levels of information about the underlying system. However, resource limitations and computational power restrict the number of sensors/data that can be processed in real-time in most complex systems. These challenges necessitate the need for selecting/scheduling a subset of sensors to obtain measurements that guarantee the best monitoring objectives. This paper focuses on sensor scheduling for systems modeled by hidden Markov models. Despite the development of several sensor selection and scheduling methods, existing methods tend to be greedy and do not take into account the long-term impact of selected sensors on monitoring objectives. This paper formulates optimal sensor scheduling as a reinforcement learning problem defined over the posterior distribution of system states. Further, the paper derives a deep reinforcement learning policy for offline learning of the sensor scheduling policy, which can then be executed in real-time as new information unfolds. The proposed method applies to any monitoring objective that can be expressed in terms of the posterior distribution of the states (e.g., state estimation, information gain, etc.). The performance of the proposed method in terms of accuracy and robustness is investigated for monitoring the security of networked systems and the health monitoring of gene regulatory networks.

1. Introduction

Most real-world systems are complex and dynamic, often monitored through multiple sensors. These sensors provide diverse information about different parts of the systems. Thus, given the large size of systems and the cost and resource limitations, collecting and processing data from all sensors at all times is impossible. Hidden Markov models (HMMs) are a well-known class of statistical models that have been used in a wide range of applications, such as computer networks, systems biology, robotics, ecology, and smart grids (Glennie, Adam, Leos-Barajas, Michelot, Photopoulou and McClintock (2023); Imani, Imani and Ghoreishi (2022); Raskar and Nema (2022); Hosam (2022); Aoudni, Donald, Farouk, Sahay, Babu, Tripathi and Dhabliya (2022); Mousavi, Bevan, Kucukdemiral and Fekih (2024); Mor, Garhwal and Kumar (2020); Mustafa, Allen and Appiah (2019); Ravari, Ghoreishi and Imani (2024); Kouadri, Hajji, Harkat, Abodayeh, Mansouri, Nounou and Nounou (2020)). Several methods have been developed for state estimation and filtering of HMMs. The state estimation performance by these methods significantly relies on the information carried in data/measurements fed into these methods. In fact, measurements provided by different sensors have different levels of information about systems at different time steps. Monitoring in HMMs consists of providing the most useful measurements to the state estimation and filtering approaches. This can be achieved by selecting/scheduling a subset of sensors to obtain measurements that ensure the best monitoring objectives, such as effectively estimating the underlying system state, diagnosing abnormalities, control, etc.

Several methods have been developed for sensor scheduling in dynamical systems. These methods include a wide class of approaches developed for linear dynamical systems, such as those that aim at minimizing the error covariance of the corresponding Kalman filter (Liu, Li, Johansson, Mårtensson and Xie (2022); Han, Wu, Mo and Xie (2017)). The extension of these approaches to nonlinear dynamical systems has been achieved through extended and unscented Kalman filtering (Li, Yu, Xia and Yang (2019)). However, these approaches are applicable only to systems that can be linearized, and their stochasticity in state and measurement can be well-approximated by Gaussian noises. For nonlinear systems with non-Gaussian uncertainty, several myopic and tree search approaches have been developed for sensor scheduling (Shamaiah, Banerjee and Vikalo (2010); Vitus, Zhang, Abate, Hu and Tomlin (2012)). Additionally, a sensor scheduling method based on an adaptive grid is introduced in (Vaisenberg, Motta, Mehrotra and Ramanan (2014)) for sentient spaces, such as smart video surveillance; however, the solution is only applicable to relatively small systems.

In recent years, deep reinforcement learning (RL) techniques have also gained attention for sensor scheduling in dynamical systems. In (Leong, Ramaswamy, Quevedo, Karl and Shi (2020)), a deep RL-based sensor scheduling method is introduced for monitoring of cyber-physical systems, and a deep RL monitoring policy is developed in (Yang, Rao, Lin, Xu and Shi (2022)) for remote state estimation with limited bandwidth. Both of these applications are modeled using a linear and Gaussian state space, where Kalman

*Corresponding author
alali.m@northeastern.edu (M. Alali); kazeminajafabadi.a@northeastern.edu (.A. Kazeminajafabadi); m.imani@northeastern.edu (.M. Imani)
alali.sites.northeastern.edu (M. Alali); imani.lab.northeastern.edu (.M. Imani)
ORCID(s): 0000-0002-5458-5273 (M. Alali); 0009-0009-8174-8507 (.A. Kazeminajafabadi); 0000-0001-9570-9909 (.M. Imani)

filtering combined with the deep Q-network (DQN) approach (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland and Ostrovski (2015)) have been employed to derive monitoring solutions. However, these approaches are not applicable to the general form of HMMs with nonlinear and non-differentiable state and measurement processes and non-Gaussian noises. Finally, a double DQN sensor scheduling is developed in (Zheng, Liu, Zhang and Lan (2023)) for target tracking in underwater wireless sensor networks. However, the approach is only developed for a specific monitoring objective and application, making it impractical for a more general class of dynamical systems.

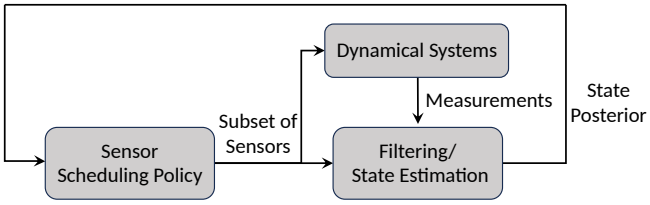

Figure 1: Schematic diagram of the proposed deep reinforcement learning sensor scheduling process.

This paper develops a deep reinforcement learning sensor scheduling policy for a general class of HMMs. The proposed policy consists of offline learning and online execution. In offline learning, the HMMs are mapped into a belief space, a continuous state space encompassing the posterior distribution of system states. We demonstrate that the belief transition is influenced by the last selected subset of sensors, and the belief transitions follow the Markov steps. We formulate the optimal sensor scheduling process as a reinforcement learning problem, with the policy defined over the belief state, the subset of sensors indicating the action space, and the one-step change in the monitoring objective (e.g., state estimation error) as the reward function. Since the exact solution to the sensor scheduling problem is not achievable due to the continuous nature of belief space, this paper approximates the optimal sensor scheduling solution using the combination of deep reinforcement learning and Bayesian filtering. The sensor scheduling policy, represented by a deep neural network, is trained according to a sequence of trajectories simulated from HMMs before the start of the execution.

Fig. 1 represents the schematic diagram related to the execution process of the proposed policy. Given the current posterior of states, the proposed policy selects a subset of sensors that measurements should be collected from in the next time step. Then, the sensor subset and the measurements are fed into a state estimation method. The state posterior distribution (often computed by the state estimation methods) is used to select the subset of sensors in the next time step. The proposed policy ensures effective scheduling of the subsets of sensors as new observations become available in order to achieve the best monitoring performance (e.g., most accurate state estimation). In this paper, the performance of the proposed policy is assessed using two different problems: 1) security monitoring of computer networks; 2) health monitoring of gene regulatory networks.

Two key contributions of the proposed policy, differentiating it from existing methods, are: 1) applicability to a general class of hidden Markov models, without constraint to the linearity of processes or Gaussianity of the noise characteristics; 2) generalizable monitoring solutions for any arbitrary monitoring objectives that can be expressed through the posterior distribution of states.

The remaining sections of this paper are structured as follows. Section 2 presents a detailed description of the hidden Markov models and the proposed sensor scheduling policy. Section 3 provides full details of the deep reinforcement learning sensor scheduling policy and its algorithm. Further, Section 4 includes the numerical experiments and Section 5 contains conclusions and a discussion of limitations.

2. Proposed Sensor Scheduling Policy

2.1. Hidden Markov Models

Hidden Markov models (HMMs) are a popular class of models for representing the dynamic behavior of complex systems observed through time-series data (Glennie et al. (2023); Raskar and Nema (2022); Hosam (2022); Aoudni et al. (2022); Mor et al. (2020); Mustafa et al. (2019); Kouadri et al. (2020)). The underlying dynamics of these systems are represented using a state process. This can be expressed as:

$$\mathbf{x}_k \sim p(. \mid \mathbf{x}_{k-1}), \tag{1}$$

where $p(.)$ denotes a probability mass function and $\mathbf{x}_k$ denotes the system's state at a given time step k. This paper assumes that the state space is finite with n elements denoted by $\mathcal{X} = (\mathbf{x}^1, \dots, \mathbf{x}^n)$. Note that the control inputs, if available, could also be incorporated into the state process (e.g., $p(. \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1})$, where $\mathbf{a}_{k-1}$ is the control input at time step $k-1$).

We assume that $\mathcal{S}$ contains all subsets of sensors in which a subset can be selected at any given time for monitoring the system. Each subset often provides information about part of the system. Letting $\mathbf{x}_k$ be the true unobserved state of the system at time step k, this state can only be partially observed through any selected subset $\mathbf{s}_{k-1}$ as:

$$\mathbf{y}_k \sim p(. \mid \mathbf{s}_{k-1}, \mathbf{x}_k), \tag{2}$$

where the measurement process provides a nonlinear and probabilistic realization of the underlying system state. For instance, in network security, subsets can refer to a set of computers on which the firewall can run to check for possible compromises. Considering the network structure and all available information, monitoring should be conducted effectively, as monitoring certain nodes might not provide enough information about the entire network compromises and could jeopardize the network security. This paper assumes that the state and measurement processes in (1) and (2) are fully known. Future work will study developing sensor scheduling policies for partially known and unknown systems.

2.2. Sensor Scheduling for Optimal Monitoring

The optimal monitoring problem in a system modeled by HMMs consists of selecting the sequence of sensor subsets that maximize the performance of monitoring objectives. One such objective is accurately estimating the state of the system, which can be measured through the posterior distribution of the system state. Given $\mathbf{s}_{0:r-1} = (\mathbf{s}_0, ..., \mathbf{s}_{r-1})$ be the sequence of the sensors and $\mathbf{y}_{1:r} = (\mathbf{y}_1, ..., \mathbf{y}_r)$ be the observed data up to time step r, the posterior distribution of the system state given the information up to time step r can be expressed as:

$$p(\mathbf{x}_r = \mathbf{x}^i \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r}), \text{ for } i = 1, ..., n. \tag{3}$$

If this posterior is peaked over a single state, then it indicates the rich knowledge provided by the selected sensors about the underlying system. On the other hand, if multiple states have large probabilities in the posterior, then the selected sensors do not clearly help to identify the true unobserved system state.

The information carried in the posterior distribution of state given the sequence of selected sensors $\mathbf{s}_{0:r-1}$ and observed data $\mathbf{y}_{1:r}$ can be expressed using the following entropy measure:

$$\begin{aligned}\mathcal{H}[p(\mathbf{x}_r \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r})] = -\sum_{i=1}^{n} p(\mathbf{x}_r = \mathbf{x}^i \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r}) \\ \times \log p(\mathbf{x}_r = \mathbf{x}^i \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r}),\end{aligned} \tag{4}$$

where $0 \leq \mathcal{H}[.] \leq -\log \frac{1}{n}$ indicates the entropy. $\mathcal{H}[.]$ close to zero represents cases with distribution peaked over a single state and, consequently, a more confident knowledge of the true underlying system state.

Before any selection and observing any data, if the goal is to select a sequence of sensors for maximum information gain, the optimal monitoring can be expressed by minimizing the entropy of the conditional state distributions as:

$$\mathbf{s}^*_{0:k-1} = \underset{\mathbf{s}_{0:k-1} \in \mathcal{S}^k}{\operatorname{argmin}} \sum_{r=1}^{k} \mathbb{E}\left[\mathcal{H}[p(\mathbf{x}_r \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r})] \mid \mathbf{s}_{0:r-1}, \mathbf{y}_{1:r}\right], \tag{5}$$

where the expectation is with respect to the unobserved data given any sequence of sensors. Solving the optimization in (5) can be challenging in practice and, if obtained, cannot be deployed for real-time sensor selection. In the following paragraphs, the optimal sequential sensor scheduling strategy is formulated.

2.3. Belief State Formulation

Belief state definition: We define the belief state as the posterior distribution of the system state. The belief state is a vector of size n, which is a sufficient statistic for representing the history of information as a compact form in a Markov decision process (MDP). The belief at time k can be represented as:

$$\mathbf{b}_k(i) = p(\mathbf{x}_k = \mathbf{x}^i \mid \mathbf{s}_{0:k-1}, \mathbf{y}_{1:k}), \text{ for } i = 1, ..., n. \tag{6}$$

The initial belief $\mathbf{b}_0(i) = p(\mathbf{x}_0 = \mathbf{x}^i)$, for $i = 1, \ldots, n$, demonstrates the initial state distribution.

Upon selection of the sensor subset $\mathbf{s}_k$ at belief state $\mathbf{b}_k$ and observing the new data $\mathbf{y}_{k+1}$, the belief state $\mathbf{b}_{k+1}$ can be computed according to the Bayesian filtering (Särkkä (2013)) as:

$$\begin{aligned}\mathbf{b}_{k+1}(i) &= p(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k+1}) \\ &= \frac{p(\mathbf{y}_{k+1}, \mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k})}{\sum_{j=1}^{n} p(\mathbf{y}_{k+1}, \mathbf{x}_{k+1} = \mathbf{x}^j \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k})},\end{aligned} \tag{7}$$

for $i = 1, ..., n$, where

$$\begin{aligned}&p(\mathbf{y}_{k+1}, \mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k}) \\ &= p(\mathbf{y}_{k+1} \mid \mathbf{s}_{0:k}, \mathbf{x}_{k+1} = \mathbf{x}^i, \mathbf{y}_{1:k}) p(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k}) \\ &= p(\mathbf{y}_{k+1} \mid \mathbf{s}_k, \mathbf{x}_{k+1} = \mathbf{x}^i) p(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k}) \\ &= p(\mathbf{y}_{k+1} \mid \mathbf{s}_k, \mathbf{x}_{k+1} = \mathbf{x}^i) \\ &\qquad \times \sum_{j=1}^{n} p(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{x}_k = \mathbf{x}^j) p(\mathbf{x}_k = \mathbf{x}^j \mid \mathbf{s}_{0:k}, \mathbf{y}_{1:k}) \\ &= p(\mathbf{y}_{k+1} \mid \mathbf{s}_k, \mathbf{x}_{k+1} = \mathbf{x}^i) \sum_{j=1}^{n} p(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{x}_k = \mathbf{x}^j) \mathbf{b}_k(j).\end{aligned} \tag{8}$$

For ease of notation, we define the *prediction matrix* and the *update matrix* in the following paragraph. The *prediction matrix* M of dimension $n \times n$ corresponds to the transition matrix of the system as:

$$(M)_{ij} = p\left(\mathbf{x}_{k+1} = \mathbf{x}^i \mid \mathbf{x}_k = \mathbf{x}^j\right), \tag{9}$$

for $i, j = 1, \ldots, n$. Furthermore, given that a subset of sensors $\mathbf{s}_k$ is selected and the value of the observation vector $\mathbf{y}_{k+1}$ is provided at time $k+1$, the *update matrix* $T(\mathbf{y}_{k+1}, \mathbf{s}_k)$ of dimension $n \times n$ can be defined as a diagonal matrix with its diagonal elements as:

$$\left(T(\mathbf{y}_{k+1}, \mathbf{s}_k)\right)_{ii} = p\left(\mathbf{y}_{k+1} \mid \mathbf{s}_k, \mathbf{x}_{k+1} = \mathbf{x}^i\right), \tag{10}$$

for $i = 1, \ldots, n$. Given the definitions of the prediction matrix and the update matrix, we can re-write equation (7) as:

$$\mathbf{b}_{k+1} = \frac{T(\mathbf{y}_{k+1}, \mathbf{s}_k)\, M\, \mathbf{b}_k}{||T(\mathbf{y}_{k+1}, \mathbf{s}_k)\, M\, \mathbf{b}_k||_1}, \tag{11}$$

where $||.||_1$ represents the absolute sum of the elements of the vector. It can be seen that the belief state at time step $k+1$ given the observed data $\mathbf{y}_{k+1}$ from sensor $\mathbf{s}_k$ is only dependent on the previous belief state at time step k, and not the prior belief states. This represents that the belief state follows a Markov process.

Belief state transition: Let $\mathbf{b}$ be the current belief state, and $\mathbf{s}$ be the selected sensor for the next data. If data $\mathbf{y}$ is observed, the next belief state can be obtained using (11) as:

$$\mathbf{b}_{\mathbf{s},\mathbf{y}} = \frac{T(\mathbf{y}, \mathbf{s})\, M\, \mathbf{b}}{||T(\mathbf{y}, \mathbf{s})\, M\, \mathbf{b}||_1}. \tag{12}$$

Using (12), upon selecting the sensor $\mathbf{s}$ and prior to observing the next data $\mathbf{y}$, the belief transition from belief state $\mathbf{b}$ to $\mathbf{b}'$ can be expressed as:

$$\begin{aligned} p(\mathbf{b}' \mid \mathbf{b}, \mathbf{s}) &= \sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y} \mid \mathbf{b}, \mathbf{s}) 1_{\mathbf{b}'=\mathbf{b}_{\mathbf{s},\mathbf{y}}} \\ &= \sum_{\mathbf{y}\in\mathcal{Y}} \|T(\mathbf{y}, \mathbf{s})\, M\, \mathbf{b}\|_1\, 1_{\mathbf{b}'=\mathbf{b}_{\mathbf{s},\mathbf{y}}}, \end{aligned} \tag{13}$$

where $\mathcal{Y}$ corresponds to the space of all possible observations, and $1_{\mathbf{b}'=\mathbf{b}_{\mathbf{s},\mathbf{y}}}$ is an indicator function that returns 1 if $\mathbf{b}' = \mathbf{b}_{\mathbf{s},\mathbf{y}}$ and 0 otherwise.

Therefore, at any given belief state, upon the selection of a subset of sensors, there will be several possible next belief states before observing the next data. Note that the possible next belief states will vary with different selections of sensor subsets. This allows us to propagate the uncertainty in the belief and analyze which selections could lead to future beliefs that are more desirable according to the monitoring objective. It should be noted that the stochastic belief transitions are fully known for any HMMs with known state and measurement processes represented in (9) and (10).

Monitoring in belief state: We measure the immediate gain in the monitoring performance through the reward function defined in the belief space. For a special case that the reduction in the state distribution entropy is desired (i.e., Bayesian state estimation), the reward function can be expressed as:

$$R(\mathbf{b}, \mathbf{s}, \mathbf{b}') = \mathcal{H}[\mathbf{b}] - \mathcal{H}[\mathbf{b}'], \tag{14}$$

where $\mathbf{b}'$ is the next belief state upon selecting the sensor subset $\mathbf{s}$ at the belief state $\mathbf{b}$. The reward function can also be defined for enhancing the performance of state estimators (Li et al. (2019)). For the special case of the maximum a posteriori (MAP) state estimator, the increase in the posterior probability of the estimated state (i.e., the state with the maximum posterior probability) measures the gain in the monitoring performance as:

$$R(\mathbf{b}, \mathbf{s}, \mathbf{b}') = \max_{i\in\{1,..,n\}} \mathbf{b}'(i) - \max_{i\in\{1,..,n\}} \mathbf{b}(i). \tag{15}$$

We refer to this as the point-based reward function since it measures the changes in the maximum posterior probability of states. If the subsets of sensors have varied computational/monitoring costs, this can also be incorporated into the reward function. Aside from two aforementioned reward functions, any monitoring objectives that can be expressed through the posterior distribution of the state (i.e., belief) can be formulated as reinforcement learning problems through proper reward functions. For instance, if the goal is to effectively estimate the state of specific parts of the system (e.g., the operational servers in a network), the reward function can be defined in terms of the change in the belief state corresponding to those system parts.

2.4. Optimal Sensor Scheduling

We define a deterministic sensor scheduling policy $\pi : \mathcal{B} \to \mathcal{S}$ as mapping any belief state in the belief space to a subset of sensors. The expected discounted return at belief state $\mathbf{b} \in \mathcal{B}$, after selecting a subset of sensors $\mathbf{s}$ from the set $\mathcal{S}$, and subsequently following policy π, can be defined as the following state-value function:

$$V^{\pi}(\mathbf{b}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1}) \mid \mathbf{b}_0 = \mathbf{b}, \mathbf{s}_{1:\infty} \sim \pi\right], \tag{16}$$

for $\mathbf{b} \in \mathcal{B}$; where γ is the discount factor, which takes values between 0 and 1, and the expectation is computed taking into account the uncertainty associated with the belief transition. The discount factor specifies the importance of future rewards compared to early staged rewards. The state-action value function for the policy π can also be defined as:

$$Q^{\pi}(\mathbf{b}, \mathbf{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1}) \,|\, \mathbf{b}_0 = \mathbf{b}, \mathbf{s}_0 = \mathbf{s}, \mathbf{s}_{1:\infty} \sim \pi\right], \tag{17}$$

for $\mathbf{b} \in \mathcal{B}$ and $\mathbf{s} \in \mathcal{S}$. The optimal policy holds the maximum state value and state-action value function as: $\pi^* = \text{argmax}_{\pi} V^{\pi}(\mathbf{b})$ and $\pi^* = \text{argmax}_{\pi} Q^{\pi}(\mathbf{b}, \mathbf{s})$, for $\mathbf{b} \in \mathcal{B}, \mathbf{s} \in \mathcal{S}$. Note that the optimal state value function V^* and state-action value function Q^* correspond to the optimal policy π^*. For any arbitrary value function (V), the Bellman equation can be written as:

$$\begin{aligned} \mathcal{T}^*[V](\mathbf{b}) &= \max_{\mathbf{s}\in\mathcal{S}} \mathbb{E}_{\mathbf{b}'|\mathbf{b},\mathbf{s}}\left[R(\mathbf{b}, \mathbf{s}, \mathbf{b}') + \gamma\, V(\mathbf{b}')\right] \\ &= \max_{\mathbf{s}\in\mathcal{S}} \sum_{\mathbf{b}'\in\mathcal{B}} p(\mathbf{b}' \mid \mathbf{b}, \mathbf{s}) \Big(R(\mathbf{b}, \mathbf{s}, \mathbf{b}') + \gamma\, V(\mathbf{b}') \Big), \end{aligned} \tag{18}$$

for $\mathbf{b} \in \mathcal{B}$; where $\mathcal{T}^*$ is the Bellman optimality operator, and the second line utilizes the belief transition in (13). The Bellam optimality operator is a γ-contractive in the L_∞-norm for any MDP (Szepesvári (2010)), and its fixed-point solution corresponds to the optimal value function. The fixed-point solution could be obtained by starting from any arbitrary V_0 and repeatedly applying $V_{t+1}(\mathbf{b}) = \mathcal{T}^*[V_t](\mathbf{b})$ for all $\mathbf{b} \in \mathcal{B}$, and $t = 0, 1, ...$, until a fixed point solution for the value is reached. However, performing the Bellman operator for all the belief states $\mathbf{b} \in \mathcal{B}$ is not possible, as the belief space is a large and continuous space. The belief space is an n-simplex (Δ_n), and as the state space size (n) increases, the belief space size greatly increases. Therefore, dynamic programming approaches such as value iteration or policy iteration (Sutton and Barto (2018)) methods cannot be employed to find the optimal sensor scheduling policy due to the continuity of belief space. Our proposed deep reinforcement learning approach to approximate the optimal policy is described in the next section.

3. Deep Reinforcement Learning Sensor Scheduling Policy

In this paper, we employ a deep reinforcement learning method for obtaining the sensor scheduling policy over a

large belief space. The sensor scheduling problem contains a large belief space with a limited number of actions (i.e. subset of sensors). Therefore, the deep Q-network (DQN) method (Mnih et al. (2015)) is an ideal candidate for solving this problem. The state-action value function in (17) is represented by two fully connected feed-forward deep neural networks, which share similar structures (input, output, layers, and neurons). The first network is called Q-network, denoted by $Q_{\mathbf{w}}$, and the second network is called target-network, indicated by $Q_{\mathbf{w}^-}$, where $\mathbf{w}$ and $\mathbf{w}^-$ denote the weights of the deep neural networks. The input to these networks is the belief state $\mathbf{b}$, which is a vector of size n. The outputs of these neural networks represent the Q-values associated with all the sensor subsets, i.e., $Q_{\mathbf{w}}(\mathbf{b}, \mathbf{s}^1), ..., Q_{\mathbf{w}}(\mathbf{b}, \mathbf{s}^{|\mathcal{S}|})$ for Q-network. The initial weights for Q-network and target-network are set randomly.

Let $\mathcal{D}$ be a replay memory, which is used for storing the history of the beliefs, subsets of sensors, and rewards during the training. The training process consists of multiple episodes, where each episode starts with an initial belief state $\mathbf{b}_0$ (if known) or a random belief sample from the belief space (if unknown). At step t of each episode, the Q-network is used for selecting the next sensor subset. Using the latest Q-network, the Q-value corresponding to all subsets of sensors can be computed from the belief state $\mathbf{b}_t$ as $Q_{\mathbf{w}}(\mathbf{b}_t, \mathbf{s})$, for all $\mathbf{s} \in \mathcal{S}$. The sensor subset is then selected using the computed Q-values and the exploratory epsilon-greedy policy (Sutton and Barto (2018)) as:

$$\mathbf{s}_t \sim \begin{cases} \text{argmax}_{\mathbf{s}\in\mathcal{S}}\, Q_{\mathbf{w}}(\mathbf{b}_t, \mathbf{s}) & \text{w.p. } 1-\epsilon \\ \text{Random}\{\mathbf{s}^1, ..., \mathbf{s}^{|\mathcal{S}|}\} & \text{w.p. } \epsilon \end{cases}, \tag{19}$$

where the operator "$\sim$" indicates that $\mathbf{s}_t$ is a sample drawn from the distribution on the right, and $0 \leq \epsilon \leq 1$ is the epsilon-greedy policy rate, which controls the level of exploration during the learning process. The right distribution is categorical distribution, where the subset of sensors with the maximum Q-value (i.e., $\text{argmax}_{\mathbf{s}\in\mathcal{S}}\, Q_{\mathbf{w}}(\mathbf{b}_t, \mathbf{s})$) has the probability of $(1-\epsilon+\epsilon/|\mathcal{S}|)$ and the other subsets of sensors have the probability of $\epsilon/|\mathcal{S}|$. Since ϵ is often small, the subset with the maximum Q-value has the largest probability of being selected.

Upon selecting $\mathbf{s}_t$, the next belief state is a sample from $p(. \mid \mathbf{b}_t, \mathbf{s}_t)$. This sample can be generated by first drawing a sample from the current belief as:

$$l \sim \text{Cat}\left((M\mathbf{b}_t)_1, ..., (M\mathbf{b}_t)_n\right), \tag{20}$$

where $(M\mathbf{b}_t)_i$ refers to the ith component of the $M\mathbf{b}_t$ vector and "Cat" stands for categorical distribution. A sample from the next measurement can be obtained as $\mathbf{y}_{t+1} \sim p(\mathbf{y} \mid \mathbf{x}_{t+1} = \mathbf{x}^l, \mathbf{s}_t)$. Then, using (11), the next belief, $\mathbf{b}_{t+1}$, can be obtained.

Upon observing this transition, one can compute the reward function using (14) or (15) as:

$$r_t = R(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1}). \tag{21}$$

The created $(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1}, r_t)$ at each step of the episode is saved at the end of the replay memory and replaces the oldest experience if it is full.

The Q-network $Q_{\mathbf{w}}$ weights need to be updated every $N_{\text{Q\_Update}}$ steps. This can be done by randomly selecting a minibatch from the experiences in the replay memory $\mathcal{D}$. Let the selected minibatch be denoted by:

$$Z = \{(\mathbf{b}_i, \mathbf{s}_i, \mathbf{b}_{i+1}, r_i)\}_{i=1}^{N_{\text{batch}}} \sim \mathcal{D}, \tag{22}$$

where N_{batch} is the size of the minibatch. For each selected experience, the target-network, $Q_{\mathbf{w}^-}$, can be used to calculate the following target values:

$$z_i = r_i + \gamma \max_{\mathbf{s}\in\mathcal{S}} Q_{\mathbf{w}^-}(\mathbf{b}_{i+1}, \mathbf{s}), \tag{23}$$

for $i = 1, ..., N_{\text{batch}}$. Using the created target values for the minibatch, the objective is to update the Q-network weights by minimizing the following mean squared error loss:

$$L(Z; \mathbf{w}, \mathbf{w}^-) = \sum_{i=1}^{N_{\text{batch}}} \left(z_i - Q_{\mathbf{w}}(\mathbf{b}_i, \mathbf{s}_i)\right)^2. \tag{24}$$

The minimization of this loss function can be achieved using a stochastic gradient optimization approach such as Adam (Kingma and Ba (2015)), where back-propagation methods compute the gradient of the loss function. This can be expressed as:

$$\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} L(Z; \mathbf{w}, \mathbf{w}^-), \tag{25}$$

where η is the learning rate, and L represents the loss in (24).

Upon updating the Q-network weights $\mathbf{w}$, the weights of the target network, $\mathbf{w}^-$, should be updated in a soft format as:

$$\mathbf{w}^- = (1-\tau)\mathbf{w}^- + \tau\mathbf{w}, \tag{26}$$

where τ is the soft update hyperparameter (often close to 0.001). This slow approach of the target network weights to the Q-network is a key aspect of the stability of the deep reinforcement learning process. At each step, new experiences will be added to the replay memory, and an update takes place every $N_{\text{Q\_Update}}$ steps. The process continues until a fixed number of episodes or a pre-specified performance is achieved (stopping criteria).

Upon termination of the training process, the policy at any given belief state $\mathbf{b}$ can be computed using the latest Q-network as:

$$\mathbf{s}^* \approx \underset{\mathbf{s}\in\mathcal{S}}{\text{argmax}}\; Q_{\mathbf{w}}(\mathbf{b}, \mathbf{s}). \tag{27}$$

This can be seen as a greedy mode of the epsilon-greedy policy in (19), where no further exploration is needed upon the termination of the training process. All the steps of the proposed deep reinforcement learning sensor scheduling policy are summarized in Algorithm 1.

Algorithm 1 The proposed sensor scheduling policy for effective monitoring of dynamical systems.

Number of system states n, initial belief $\mathbf{b}_0(i) = p(\mathbf{x}_0 = \mathbf{x}^i)$, for $i = 1, \dots, n$ or equivalently prior distribution of the states, possible subsets of sensors $\mathcal{S} = \{\mathbf{s}^1, ..., \mathbf{s}^L\}$, horizon T.
Size of replay memory $|\mathcal{D}|$, length of batch sample N_{batch}, length of episodes T_{episode}, the training step $N_{\text{Q\_Update}}$, discount factor γ, learning rate η, epsilon-greedy hyperparameter ϵ, soft update parameter τ, Q-network and target network with random initial weights $\mathbf{w}$ and $\mathbf{w}^-$.
Replay Memory $\mathcal{D} = \{\}$, counter $= 0$.
while (stopping criteria is not met) **do**
    Initial belief $\mathbf{b}_0$.
    for $t = 0$ to T_{episode} **do**
        counter $=$ counter $+ 1$.
        Generate $\mathbf{s}_t$ at belief state $\mathbf{b}_t$ from epsilon-greedy policy — Eq. (19).
        Sample a possible next state $\mathbf{x}^l$ — Eq. (20)
        Obtain a sample from the next measurement as $\mathbf{y}_{t+1} \sim p(\mathbf{y} \mid \mathbf{x}_{t+1} = \mathbf{x}^l, \mathbf{s}_t)$.
        Calculate the next belief using $\mathbf{b}_{t+1} = \frac{T(\mathbf{y}_{t+1}, \mathbf{s}_t)\, M\, \mathbf{b}_t}{||T(\mathbf{y}_{t+1}, \mathbf{s}_t)\, M\, \mathbf{b}_t||_1}$.
        Use $\mathbf{b}_{t+1}$ to compute the reward based on the objective of the problem $r_t = R(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1})$ — Eq. (14) or (15).
        Save experience $(\mathbf{b}_t, \mathbf{s}_t, \mathbf{b}_{t+1}, r_t)$ into $\mathcal{D}$; if it is full, replace with the oldest experience.
        if counter $= N_{\text{Q\_Update}}$ **then**
            Generate N_{batch} random samples from $\mathcal{D}$ — Eq. (22).
            Update the Q-network using Eqs. (23)-(24).
            Update the target network using Eq. (26).
            counter $= 0$.
        end if
    end for
end while
Use the latest Q-network for sensor scheduling policy — Eq. (27).

4. Numerical Experiments

In this section, the performance of the proposed sensor scheduling policy is assessed through numerical experiments for security monitoring of computer networks and health monitoring of gene regulatory networks. The results are averaged across 1000 trials using the following parameters: number of hidden layers 3, number of neurons in hidden layers 128, $\eta = 5 \times 10^{-4}$, $|\mathcal{D}| = 10^5$, $N_{\text{batch}} = 64$, $\gamma = 0.95$, $\epsilon = 0.1$, $N_{\text{Q\_Update}} = 4$, and $\tau = 10^{-3}$.

The performance of the proposed method is compared with the expected information gain (EIG), which is a well-known class of approaches for monitoring complex systems (Kreucher (2005); Williams (2007)). The EIG policy aims to sequentially select a subset of sensors at each step to maximally reduce the entropy in the posterior distribution of state as:

$$\mathbf{s}_k = \underset{\mathbf{s}\in\mathcal{S}}{\operatorname{argmax}}\, \mathbb{E}\Big[\mathcal{H}\left[p(\mathbf{x}_k \,|\, \mathbf{s}_{0:k-1}, \mathbf{y}_{1:k})\right] - \mathcal{H}\left[p(\mathbf{x}_{k+1} \,|\, \mathbf{s}_{0:k-1}, \mathbf{s}_k = \mathbf{s}, \mathbf{y}_{1:k+1})\right] \mid \mathbf{y}_{1:k}, \mathbf{s}_{0:k-1}, \mathbf{s}_k = \mathbf{s}\Big], \tag{28}$$

where the expectations are with respect to the unobserved data at time step $k + 1$. The second comparison method is the maximum a posteriori (MAP) policy (Shamaiah et al. (2010)). This established method sequentially selects sensor subsets as:

$$\mathbf{s}_k = \underset{\mathbf{s}\in\mathcal{S}}{\operatorname{argmax}}\, \mathbb{E}\Big[\max_{i\in\{1,\dots,n\}} p(\mathbf{x}_{k+1} = \mathbf{x}^i \,|\, \mathbf{s}_{0:k-1}, \mathbf{s}_k = \mathbf{s}, \mathbf{y}_{1:k+1}) - \max_{i\in\{1,\dots,n\}} p(\mathbf{x}_k = \mathbf{x}^i \,|\, \mathbf{s}_{0:k-1}, \mathbf{y}_{1:k}) \mid \mathbf{y}_{1:k}, \mathbf{s}_{0:k-1}, \mathbf{s}_k = \mathbf{s}\Big]. \tag{29}$$

The MAP policy outlined in (29) follows a greedy strategy designed to maximize monitoring performance in a single step. Conversely, our proposed method strives to maximize the posterior probability of states throughout the entire time horizon. The MAP policy can be seen as a greedy strategy with the point-based reward function in (15), whereas the EIG policy in (28) corresponds to the greedy version of the proposed policy with entropy reward function.

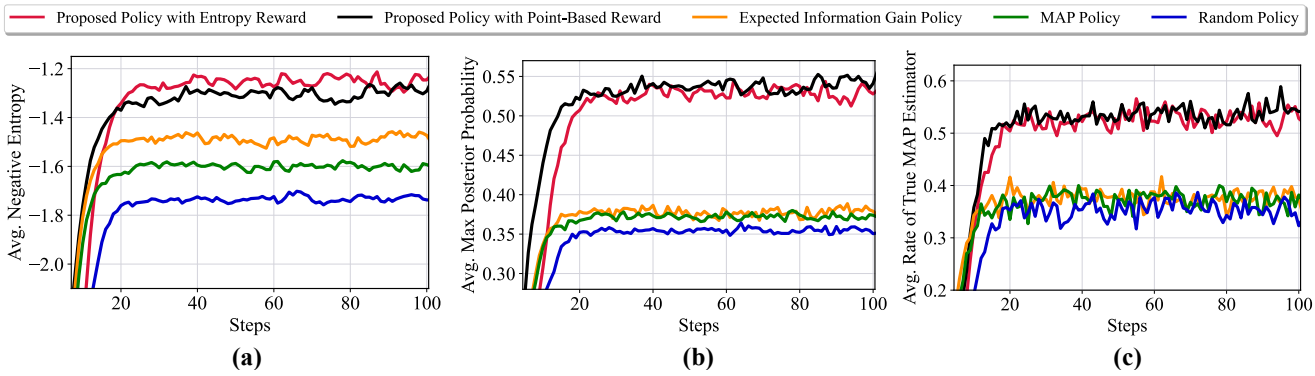

Figure 2: Performance comparison of different sensor scheduling techniques: (a) average negative entropy (b) average maximum posterior probability of the states (c) average rate of true MAP estimator.

4.1. Monitoring Network Systems for Security

Bayesian attack graphs (BAGs) are a well-known class of models that represent the propagation of attacks and compromises in networks (Kordy, Pièvre-Cambacédès and Schweitzer (2014)). They can be seen as a Markov process with binary state variables. The state vector, denoted as $\mathbf{x}_k = [\mathbf{x}_k(1), ..., \mathbf{x}_k(d)]$, captures the compromised status of all (d) nodes in the network. The ith state variable takes values 1 or 0, where 1 and 0 represent the existence and lack of compromise at node i, respectively. Meanwhile, the binary values for each variable lead to $n = 2^d$ possible states, indicating all possible compromise status in the network.

BAGs model cybersecurity vulnerabilities via exploit probabilities (ρ_{ij}) between nodes, along with probabilities representing the external network vulnerability (ρ_i). The ρ_{ij} shows the likelihood of attack progression from node i to j. Given distinct node characteristics and network defender's actions $(\mathbf{a}_{k-1})$ to remove compromises, the conditional probability of the jth node's compromise at time step k, based on its state $(\mathbf{x}_{k-1}(j))$ in the previous step $(k-1)$, can be expressed for AND and OR nodes as:

- **AND Nodes**:

$$p(\mathbf{x}_k(j) = 1 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1}) = \begin{cases} (1_{j\notin\mathbf{a}_{k-1}} + \beta 1_{j\in\mathbf{a}_{k-1}}) \left[\rho_j + (1-\rho_j) \prod_{i\in D_j} \rho_{ij}^{1_{\mathbf{x}_{k-1}(i)=1}}\right] & \text{if } \mathbf{x}_{k-1}(j) = 0, \\ 1_{j\notin\mathbf{a}_{k-1}} + \beta 1_{j\in\mathbf{a}_{k-1}} & \text{if } \mathbf{x}_{k-1}(j) = 1, \end{cases} \tag{30}$$

- **OR Nodes**:

$$p(\mathbf{x}_k(j) = 1 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1}) = \begin{cases} (1_{j\notin\mathbf{a}_{k-1}} + \beta 1_{j\in\mathbf{a}_{k-1}}) \left[\rho_j + (1-\rho_j) \left[1 - \prod_{i\in D_j} \left(1 - \rho_{ij}^{1_{\mathbf{x}_{k-1}(i)=1}}\right)\right]\right] & \\ & \text{if } \mathbf{x}_{k-1}(j) = 0, \\ 1_{j\notin\mathbf{a}_{k-1}} + \beta 1_{j\in\mathbf{a}_{k-1}} & \text{if } \mathbf{x}_{k-1}(j) = 1, \end{cases} \tag{31}$$

where β is the probability of effectively eliminating compromises within the system's nodes, D_j is the set of all possible attacks from neighboring nodes of j, $1_{\text{condition}}$ returns 1 if the condition is true, and 0 otherwise. Note that the conditional probabilities in (30) and (31) consider both the external (i.e., ρ_j) and internal (i.e., ρ_{ij}) attacks.

We investigate sensor scheduling on a 10-node BAG shown in Fig. 3. The network's vulnerabilities are captured by ρ_{ij} values: $\rho_{12} = 0.7, \rho_{14} = 0.6, \rho_{25} = 0.6, \rho_{36} = 0.55, \rho_{39} = 0.7, \rho_{47} = 0.7, \rho_{58} = 0.7, \rho_{62} = 0.7, \rho_{87} = 0.7, \rho_{94} = 0.6, \rho_{98} = 0.7, \rho_{108} = 0.7, \rho_{109} = 0.4$, and ρ_i values: $\rho_1 = 0.65, \rho_3 = 0.6, \rho_{10} = 0.55$. Further, the probability of successfully removing a compromise by a stochastic network defender (i.e., β) is 0.5.

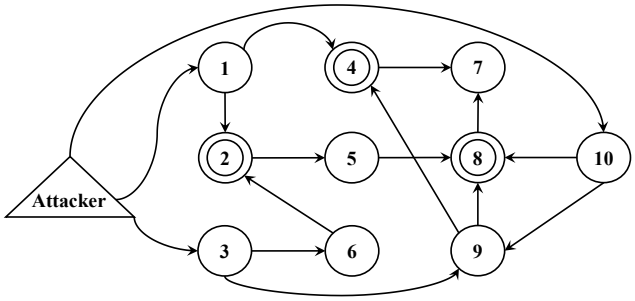

Figure 3: The BAG consisting of 10 components. Double circles indicate AND nodes, while regular circles denote OR nodes.

Monitoring is achieved by selecting a single node at each time to check its compromise, with minimal disruption to the system's operation. Therefore, the sensor space consists of $\mathcal{S} = \{\mathbf{s}^1, ..., \mathbf{s}^{10}\}$, with the ith sensor $(\mathbf{s}_{k-1} = \mathbf{s}^i)$ providing a noisy realization of the true network compromise at node i as:

$$\mathbf{y}_k = \begin{cases} 1 \text{ if } \mathbf{x}_k(i) = 1 & \text{w.p. } \alpha \\ 0 \text{ if } \mathbf{x}_k(i) = 1 & \text{w.p. } 1-\alpha \\ 0 \text{ if } \mathbf{x}_k(i) = 0 & \text{w.p. } 1 \end{cases}, i = 1, ..., 10, \tag{32}$$

where α is the measurement accuracy rate. This measurement process identifies a compromise in a compromised node with probability α and misses it with probability $1-\alpha$. In our experiments, we consider α to be 0.7. Note that the parameters of the state and measurement processes in (30)-(32) are assumed to be known.

Our proposed policy is trained for monitoring the security of the 10-node network system according to the entropy and point-based rewards in (14) and (15) and compared with

Table 1
Average entropy and average rate of true MAP estimator per step during monitoring of the 10 Node BAG.

	Average Entropy per Step				Average Rate of True MAP Estimator per Step			
	$\mathcal{S}_{\text{ALL}}$	$\mathcal{S}_{\text{OR}}$	$\mathcal{S}^{\text{R}}_{\text{OR}}$	$\mathcal{S}_{\text{AND}}$	$\mathcal{S}_{\text{ALL}}$	$\mathcal{S}_{\text{OR}}$	$\mathcal{S}^{\text{R}}_{\text{OR}}$	$\mathcal{S}_{\text{AND}}$
Proposed Policy (Entropy)	1.552 ± 0.091	$\mathbf{1.726 \pm 0.081}$	$\mathbf{1.854 \pm 0.074}$	1.639 ± 0.085	0.478 ± 0.013	0.408 ± 0.010	0.370 ± 0.008	0.402 ± 0.009
Proposed Policy (Point-Based)	$\mathbf{1.520 \pm 0.079}$	1.728 ± 0.079	1.881 ± 0.073	$\mathbf{1.629 \pm 0.081}$	$\mathbf{0.491 \pm 0.013}$	$\mathbf{0.446 \pm 0.010}$	$\mathbf{0.390 \pm 0.005}$	$\mathbf{0.416 \pm 0.008}$
EIG Policy	1.686 ± 0.077	1.856 ± 0.072	1.941 ± 0.072	1.684 ± 0.077	0.356 ± 0.007	0.348 ± 0.005	0.339 ± 0.006	0.368 ± 0.008
MAP Policy	1.802 ± 0.078	1.926 ± 0.069	1.986 ± 0.073	1.734 ± 0.074	0.349 ± 0.008	0.339 ± 0.006	0.342 ± 0.006	0.362 ± 0.008
Random Policy	1.968 ± 0.078	1.969 ± 0.072	1.985 ± 0.072	1.850 ± 0.078	0.321 ± 0.009	0.343 ± 0.004	0.337 ± 0.006	0.341 ± 0.008

the EIG, MAP, and random policies. The following measures are compared for different policies: the average negative entropy, the maximum posterior probability of states, and the rate of true MAP estimator. The rate of true MAP estimator at each time step indicates the rate at which the state with the maximum posterior probability refers to the actual system state. Let the maximum posterior probability of states and the actual state at step k be $\hat{\mathbf{x}}_k$, $\mathbf{x}^*_k$, respectively. The rate of true MAP estimator can be written as $\frac{\sum_{t=1}^{k} 1_{\hat{\mathbf{x}}_t = \mathbf{x}^*_t}}{k}$, where $1_{\hat{\mathbf{x}}_t = \mathbf{x}^*_t}$ is 1 if $\hat{\mathbf{x}}_t = \mathbf{x}^*_t$ and 0 otherwise.

Fig. 2 shows the average performance of different policies in terms of average negative entropy, the maximum posterior probability of states, and the rate of true MAP estimator. It can be seen that the proposed policies with entropy and point-based rewards exceed the performance of the other policies. More specifically, in Fig. 2(a) for average negative entropy, the proposed policy with entropy reward shows a slightly better performance in comparison to the point-based reward. The EIG policy's performance falls below the proposed policies, followed by the MAP, and random policies. For the maximum posterior probability of states, and the rate of true MAP estimator in Fig. 2(b) and (c), both of the proposed policies exhibit almost the same performance, with the performance of the point-based reward policy being slightly higher. The EIG and MAP policies in Fig. 2(b) and (c) have almost the same performance and random policy has the lowest performance.

The robustness of the proposed policy with respect to the sensor noise is considered in this part of the experiments. Fig. 4 shows the average results of all policies in terms of average rate of true MAP estimator per time step for four different measurement accuracy rates: $\alpha = 1, 0.85, 0.7, 0.5$. One can see as α decreases (i.e., sensor noise increases), the performance of all methods decreases. However, it can be seen that the proposed policies have the best performance among the others across different α values. In particular, the proposed policy with the point-based reward has performed better than the entropy reward in noisier sensor cases. Therefore, the results in Fig. 4 demonstrate the robustness of the proposed method across different noise values.

Furthermore, we investigate the impact of the monitoring subsets (i.e., sensor space) on the performance of the proposed policies. Four sensor spaces are considered: 1) all nodes: $\mathcal{S}_{\text{ALL}} = \{\mathbf{s}^1, ..., \mathbf{s}^{10}\}$ 2) all OR nodes: $\mathcal{S}_{\text{OR}} = \{\mathbf{s}^1, \mathbf{s}^3, \mathbf{s}^5, \mathbf{s}^6, \mathbf{s}^7, \mathbf{s}^9, \mathbf{s}^{10}\}$ 3) three OR nodes: $\mathcal{S}^{\text{R}}_{\text{OR}} = \{\mathbf{s}^1, \mathbf{s}^6, \mathbf{s}^7\}$ 4) all AND nodes: $\mathcal{S}_{\text{AND}} = \{\mathbf{s}^2, \mathbf{s}^4, \mathbf{s}^8\}$. Table 1

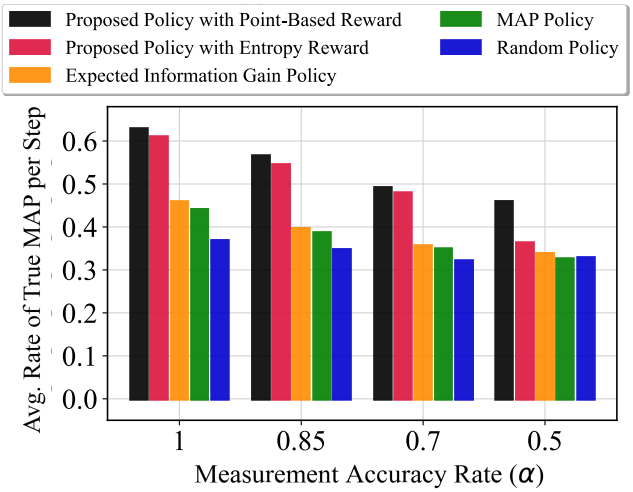

Figure 4: Performance comparison of various sensor scheduling techniques for different measurement accuracy rates.

represents the average entropy and the average rate of true MAP estimator per step obtained by various monitoring policies under different subsets. The average values in this table are accompanied with their 95% confidence intervals. Note that a single node at a time is selected for monitoring from each sensor space by all methods. For average entropy per step, the $\mathcal{S}_{\text{OR}}$ and $\mathcal{S}^{\text{R}}_{\text{OR}}$ sensor spaces achieve the best performance by the proposed policy with entropy reward. Further, the $\mathcal{S}_{\text{ALL}}$ and $\mathcal{S}_{\text{AND}}$ sensor spaces exhibit the best performance in terms of average entropy under the proposed policy with point-based reward. Also, among all sensor spaces, the best and worst average entropy per step results are achieved using $\mathcal{S}_{\text{ALL}}$ and $\mathcal{S}^{\text{R}}_{\text{OR}}$, respectively. For the average rate of true MAP estimator, one can notice that the proposed policy with the point-based reward has the best performance under all sensor space. More specifically, the sensor space with all the nodes ($\mathcal{S}_{\text{ALL}}$) has the best performance among others. The second best performance belongs to the $\mathcal{S}_{\text{OR}}$ sensor space containing 7 nodes, the third best is the $\mathcal{S}_{\text{AND}}$ space, and the worst performance is obtained by the $\mathcal{S}^{\text{R}}_{\text{OR}}$ space.

Finally, we compare the performance of the proposed sensor scheduling method with three well-known network monitoring techniques. These include a tree-based (or Monte Carlo) approach (Noel and Jajodia (2008)) that relies on the simulated most likely attack paths for the sensor selection process; the minimum mean square error (MMSE) monitoring method (Kazeminajafabadi and Imani (2023)) that

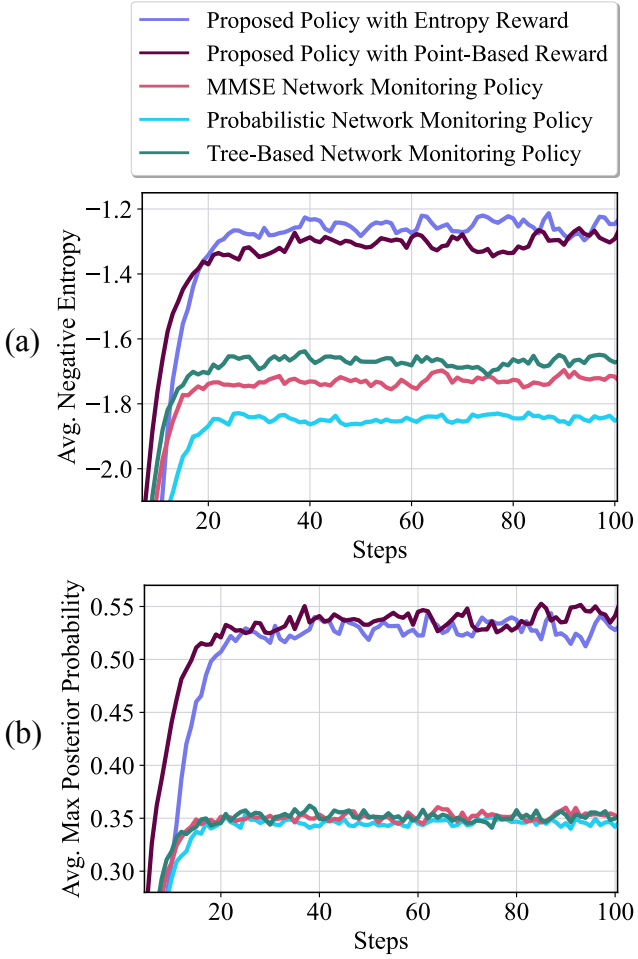

Figure 5: Performance comparison of our proposed method with three network monitoring methods: (a) average negative entropy (b) average maximum posterior probability of the states.

sequentially selects sensor subsets that hold the minimum expected mean square error in the next time step; and a probabilistic vulnerability assessment approach (Dantu, Loper and Kolan (2004)) which selects sensor subsets according to the expected increase in the probability of compromise at different nodes. Fig. 5 showcases the performance of our proposed method in comparison to these three methods. The results are presented in terms of the average negative entropy and average maximum posterior probability. Our proposed method, which incorporates entropy and point-based rewards, demonstrates considerably better performance compared to the three competing network monitoring techniques. This is due to the consideration of the system and measurement stochasticity, as well as potential compromises in the system over a longer period of time. In contrast, the other methods rely on single probable outcomes or the one-step impact of sensors in their selection processes. As expected, the proposed method with entropy reward yields the largest negative entropy values, and the proposed policy with a point-based reward function achieves the largest maximum posterior probability.

4.2. Health Monitoring of Gene Regulatory Networks

Gene regulatory networks are commonly modeled through HMMs with binary state variables (Liu and Rajapakse (2019); Hosseini and Imani (2023); Alali and Imani (2023)). We consider the mammalian cell-cycle network shown in Fig. 6 for our analysis. This network governs the cellular division process in mammalian organisms, and its dynamics hold a pivotal role in overall organismal growth. This network consists of 10 genes with the state vector at time step k represented by $\mathbf{x}_k = [\mathbf{x}_k(1), ..., \mathbf{x}_k(10)]^T$ =[CycD, Rb, p27, E2F, CycE, CycA, Cdc20, Cdh1, UbcH10, CycB]^T. Each state variable takes from a binary set $\mathbf{x}_k(i) \in \{0, 1\}$, where 1 and 0 represent the activation and inactivation of the ith gene, respectively. The state value of the 10 genes at time step k can be represented by (Imani and Braga-Neto (2017); Alali and Imani (2022)):

$$\mathbf{x}_k = \overline{\mathbf{C}\,\mathbf{x}_{k-1}} \oplus \mathbf{n}_k, \text{ for } k = 1, 2, \dots, \tag{33}$$

where $\overline{\mathbf{v}}$ maps the positive element of a vector to 1 and others to 0, "$\oplus$" indicates component-wise modulo-2 addition, $\mathbf{n}_k$ is the transition noise at time k, and $\mathbf{C}$ is the connectivity matrix of 10×10 size represented by:

$$\mathbf{C} = \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & +1 & 0 & -1 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & +1 & 0 & -1 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & +1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & +1 & +1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & +1 & 0 & +1 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & +1 \\ 0 & 0 & +1 & 0 & 0 & -1 & +1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & +1 & +1 & -1 & +1 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \end{bmatrix}. \tag{34}$$

The elements in the ith row and jth column of the connectivity matrix represent the types of regulations from gene i to gene j, where $+1$ and -1 represent the positive and negative regulations and 0 indicates no regulation. The noise process $\mathbf{n}_k$ is modeled by an independent Bernoulli random variable with intensity $p = 0.05$.

Monitoring the state value of all 10 genes at all times is often too costly or impossible. Therefore, for health monitoring, it is often critical to track the activity of a subset of genes by monitoring the state value of a single gene (or a small subset of genes). For our experiments, we consider that a single gene can be selected for monitoring at each step, which can be expressed through the sensor subset $\mathcal{S} = \{\mathbf{s}^1, ..., \mathbf{s}^{10}\}$, where $\mathbf{s}_{k-1} = \mathbf{s}^i$ indicates monitoring the state value of the ith gene as: $\mathbf{y}_k = \begin{cases} \mathbf{x}_k(i) & \text{w.p. } 0.8 \\ 1 - \mathbf{x}_k(i) & \text{w.p. } 0.2 \end{cases}$.

The performance of the proposed policy with entropy and point-based rewards alongside other comparing policies for monitoring the mammalian cell cycle network is presented in Table 2. This table shows the average entropy and average rate of true MAP estimator for different steps

Table 2
Average entropy and average maximum posterior probability during monitoring of the mammalian cell-cycle network.

	Average Entropy					Average Rate of True MAP Estimator				
	20 Steps	40 Steps	60 Steps	80 Steps	100 Steps	20 Steps	40 Steps	60 Steps	80 Steps	100 Steps
Proposed Policy (Entropy)	3.186 ± 0.007	3.195 ± 0.007	$\mathbf{3.182 \pm 0.007}$	3.189 ± 0.007	3.196 ± 0.007	0.301 ± 0.015	0.309 ± 0.015	0.324 ± 0.015	0.319 ± 0.015	0.312 ± 0.015
Proposed Policy (Point-Based)	$\mathbf{3.179 \pm 0.007}$	$\mathbf{3.189 \pm 0.007}$	3.189 ± 0.007	$\mathbf{3.180 \pm 0.007}$	$\mathbf{3.190 \pm 0.007}$	$\mathbf{0.332 \pm 0.015}$	$\mathbf{0.339 \pm 0.015}$	$\mathbf{0.326 \pm 0.015}$	$\mathbf{0.325 \pm 0.015}$	$\mathbf{0.329 \pm 0.015}$
EIG Policy	3.334 ± 0.009	3.346 ± 0.009	3.348 ± 0.009	3.327 ± 0.009	3.342 ± 0.009	0.270 ± 0.014	0.276 ± 0.014	0.263 ± 0.014	0.277 ± 0.014	0.248 ± 0.014
MAP Policy	3.252 ± 0.007	3.245 ± 0.007	3.251 ± 0.007	3.245 ± 0.007	3.239 ± 0.007	0.210 ± 0.013	0.208 ± 0.013	0.180 ± 0.012	0.241 ± 0.014	0.226 ± 0.013
Random Policy	3.449 ± 0.008	3.448 ± 0.008	3.435 ± 0.008	3.462 ± 0.008	3.436 ± 0.008	0.223 ± 0.013	0.204 ± 0.013	0.209 ± 0.013	0.195 ± 0.013	0.191 ± 0.012

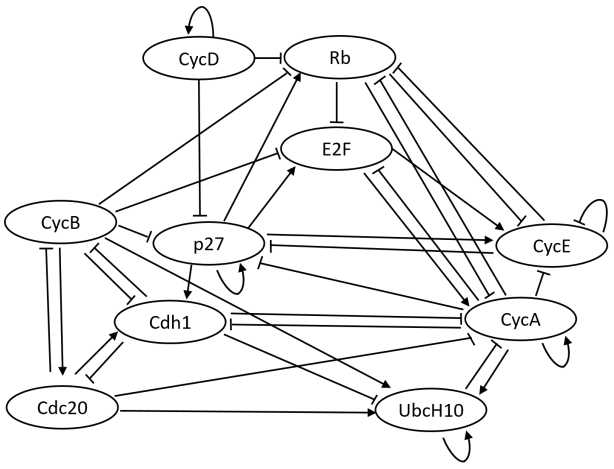

Figure 6: Pathway diagram for the mammalian cell-cycle network.

during the monitoring process. Note that the best performances of each measure at all the steps are shown with a bold color. One can see that the proposed policies yield the best performance in all the steps, which demonstrates the effectiveness of the proposed policies in monitoring the gene regulatory network. Moreover, it can be seen that, on average, the proposed policy with point-based reward has shown a better performance in comparison to the one with entropy reward in almost all the steps.

5. Conclusion

In conclusion, this paper developed a sensor scheduling policy for monitoring systems modeled by hidden Markov models (HMMs). The paper formulates optimal sensor scheduling as a reinforcement learning problem, where the actions are subsets of sensors, and the state contains the posterior distribution of system states. We developed a deep reinforcement learning sensor scheduling policy, which can be learned offline and implemented in real-time for effective multi-purpose monitoring of complex dynamical systems. The superiority of the proposed method is demonstrated in the security monitoring of networks and health monitoring of gene regulatory networks.

Two key limitations of the proposed sensor scheduling policy are the scalability and the reliance on the full system model. The former refers to the lack of scalability to large and continuous system states, and the latter indicates the need for full knowledge of state and measurement processes for training the proposed policy. Our future work will address these limitations and enable sensor scheduling to a larger class of real-world dynamical systems.

Acknowledgment

The authors acknowledge the support of the National Institute of Health award 1R21EB032480-01, National Science Foundation awards IIS-2311969 and IIS-2202395, ARMY Research Laboratory award W911NF2320179, ARMY Research Office award W911NF2110299, and Office of Naval Research award N00014-23-1-2850.

Disclosure Statement

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author, M.A., upon reasonable request.

References

Alali, M., Imani, M., 2022. Inference of regulatory networks through temporally sparse data. Frontiers in Control Engineering 3. doi:10.3389/fcteg.2022.1017256.

Alali, M., Imani, M., 2023. Reinforcement learning data-acquiring for causal inference of regulatory networks, in: American Control Conference (ACC), IEEE. doi:10.23919/ACC55779.2023.10155867.

Aoudni, Y., Donald, C., Farouk, A., Sahay, K.B., Babu, D.V., Tripathi, V., Dhabliya, D., 2022. Cloud security based attack detection using transductive learning integrated with hidden Markov model. Pattern Recognition Letters 157, 16–26. doi:10.1016/j.patrec.2022.02.012.

Dantu, R., Loper, K., Kolan, P., 2004. Risk management using behavior based attack graphs, in: International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004., pp. 445–449 Vol.1. doi:10.1109/ITCC.2004.1286496.

Glennie, R., Adam, T., Leos-Barajas, V., Michelot, T., Photopoulou, T., McClintock, B.T., 2023. Hidden Markov models: Pitfalls and opportunities in ecology. Methods in Ecology and Evolution 14, 43–56. doi:10.1111/2041-210X.13801.

Han, D., Wu, J., Mo, Y., Xie, L., 2017. On stochastic sensor network scheduling for multiple processes. IEEE Transactions on Automatic Control 62, 6633–6640. doi:10.1109/TAC.2017.2717193.

Hosam, O., 2022. An earthquake query system based on hidden Markov models. International Journal of Embedded Systems 15, 149–157. doi:10.1504/IJES.2022.123311.

Hosseini, S.H., Imani, M., 2023. Learning to fight against cell stimuli: A game theoretic perspective, in: 2023 IEEE Conference on Artificial Intelligence (CAI), IEEE. pp. 285–287. doi:10.1109/CAI54212.2023.00127.

Imani, M., Braga-Neto, U.M., 2017. Maximum-likelihood adaptive filter for partially observed Boolean dynamical systems. IEEE Transactions on Signal Processing 65, 359–371. doi:10.1109/TSP.2016.2614798.

Imani, M., Imani, M., Ghoreishi, S.F., 2022. Optimal Bayesian biomarker selection for gene regulatory networks under regulatory model uncertainty, in: 2022 American Control Conference (ACC), IEEE. pp. 1379–1385. doi:10.23919/ACC53348.2022.9867683.

Kazeminajafabadi, A., Imani, M., 2023. Optimal monitoring and attack detection of networks modeled by bayesian attack graphs. Cybersecurity 6, 22. doi:https://doi.org/10.1186/s42400-023-00155-y.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. CoRR abs/1412.6980. doi:10.48550/arXiv.1412.6980.

Kordy, B., Pièrre-Cambacédès, L., Schweitzer, P., 2014. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. Computer Science Review 13-14, 1–38. doi:10.1016/j.cosrev.2014.07.001.

Kouadri, A., Hajji, M., Harkat, M.F., Abodayeh, K., Mansouri, M., Nounou, H., Nounou, M., 2020. Hidden Markov model based principal component analysis for intelligent fault diagnosis of wind energy converter systems. Renewable Energy 150, 598–606. doi:10.1016/j.renene.2020.01.010.

Kreucher, C.M., 2005. An Information-based Approach to Sensor Resource Allocation. Phd thesis. University of Michigan. Ann Arbor, MI.

Leong, A.S., Ramaswamy, A., Quevedo, D.E., Karl, H., Shi, L., 2020. Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems. Automatica 113, 108759. doi:10.1016/j.automatica.2019.108759.

Li, L., Yu, D., Xia, Y., Yang, H., 2019. Remote nonlinear state estimation with stochastic event-triggered sensor schedule. IEEE Transactions on Cybernetics 49, 734–745. doi:10.1109/TCYB.2017.2776976.

Liu, H., Li, Y., Johansson, K.H., Mårtensson, J., Xie, L., 2022. Rollout approach to sensor scheduling for remote state estimation under integrity attack. Automatica 144, 110473. doi:10.1016/j.automatica.2022.110473.

Liu, W., Rajapakse, J., 2019. Fusing gene expressions and transitive protein-protein interactions for inference of gene regulatory networks. BMC Systems Biology 13. doi:10.1186/s12918-019-0695-x.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., 2015. Human-level control through deep reinforcement learning. Nature 518, 529. doi:10.1038/nature14236.

Mor, B., Garhwal, S., Kumar, A., 2020. A systematic review of hidden Markov models and their applications. Archives of Computational Methods in Engineering 28, 1429–1448. doi:10.1007/s11831-020-09422-4.

Mousavi, Y., Bevan, G., Kucukdemiral, I.B., Fekih, A., 2024. Observer-based high-order sliding mode control of dfig-based wind energy conversion systems subjected to sensor faults. IEEE Transactions on Industry Applications 60, 1750–1759. doi:10.1109/TIA.2023.3317823.

Mustafa, M.K., Allen, T., Appiah, K., 2019. A comparative review of dynamic neural networks and hidden Markov model methods for mobile on-device speech recognition. Neural Comput. Appl. 31, 891–899. doi:10.1007/s00521-017-3028-2.

Noel, S., Jajodia, S., 2008. Optimal ids sensor placement and alert prioritization using attack graphs. Journal of Network and Systems Management 16, 259–275. doi:10.1007/s10922-008-9109-x.

Raskar, C., Nema, S., 2022. Metaheuristic enabled modified hidden Markov model for traffic flow prediction. Computer Networks 206, 108780. doi:10.1016/j.comnet.2022.108780.

Ravari, A., Ghoreishi, S.F., Imani, M., 2024. Optimal inference of hidden Markov models through expert-acquired data. IEEE Transactions on Artificial Intellegence .

Särkkä, S., 2013. Bayesian filtering and smoothing. volume 3. Cambridge University Press. doi:10.1017/CBO9781139344203.

Shamaiah, M., Banerjee, S., Vikalo, H., 2010. Greedy sensor selection: Leveraging submodularity, in: Decision and Control (CDC), 2010 49th IEEE Conference on, IEEE. pp. 2572–2577. doi:10.1109/CDC.2010.5717225.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA, USA.

Szepesvári, C., 2010. Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers. doi:10.1007/978-3-031-01551-9.

Vaisenberg, R., Motta, A.D., Mehrotra, S., Ramanan, D., 2014. Scheduling sensors for monitoring sentient spaces using an approximate POMDP policy. Pervasive and Mobile Computing 10, 83–103. doi:10.1016/j.pmcj.2013.10.014.

Vitus, M.P., Zhang, W., Abate, A., Hu, J., Tomlin, C.J., 2012. On efficient sensor scheduling for linear dynamical systems. Automatica 48, 2482–2493. doi:10.1016/j.automatica.2012.06.092.

Williams, J.L., 2007. Information theoretic sensor management. Phd thesis. Massachusetts Institute of Technology. Cambridge, MA.

Yang, L., Rao, H., Lin, M., Xu, Y., Shi, P., 2022. Optimal sensor scheduling for remote state estimation with limited bandwidth: a deep reinforcement learning approach. Information Sciences 588, 279–292. doi:10.1016/j.ins.2021.12.043.

Zheng, L., Liu, M., Zhang, S., Lan, J., 2023. A novel sensor scheduling algorithm based on deep reinforcement learning for bearing-only target tracking in uwsns. IEEE/CAA Journal of Automatica Sinica 10, 1077–1079. doi:10.1109/JAS.2023.123159.

Mohammad Alali is a third-year Ph.D. candidate in Electrical Engineering at Northeastern University. He received his B.Sc. degree in Electrical Engineering from University of Tehran, Tehran, Iran, in 2018, and M.Sc. degree in Electrical Engineering from Montana State University, Bozeman, USA, in 2021. His research interests lie in the domains of Bayesian statistics, reinforcement learning, and experimental design. More specifically, he is currently investigating several projects with a focus on the inference, perturbation, and data collection of various Markov Models. He is the recipient of the Best Paper Finalist award from the American Control Conference in 2023.

Armita Kazeminajafabadi is a Ph.D. student in Electrical Engineering at Northeastern University. She received her B.Sc. degree in Computer Science with a minor in Mathematics from the Sharif University of Technology, Tehran, Iran, in 2022. Her research interests include cybersecurity, graphical models and machine learning.

Mahdi Imani received his Ph.D. degree in Electrical and Computer Engineering from Texas A&M University, College Station, TX in 2019. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Northeastern University. His research interests include machine learning, Bayesian statistics, and decision theory, with a wide range of applications from computational biology to cyber-physical systems. He is the recipient of several awards, including the NIH NIBIB Trailblazer award in 2022, the Oracle Research Award in 2022, the NSF CISE Career Research Initiation Initiative award in 2020, the Association of Former Students Distinguished Graduate Student Award for Excellence in Research-Doctoral in 2019, and the Best Paper Finalist award from the American Control Conference in 2023 and the 49th Asilomar Conference on Signals, Systems, and Computers in 2015.