Scalable and Decentralized Algorithms for Anomaly Detection via Learning-Based Controlled Sensing

Joseph, Geethu; Zhong, Chen; Gursoy, M. Cenk; Velipasalar, Senem; Varshney, Pramod

DOI
10.1109/TSIPN.2023.3313818

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Signal and Information Processing over Networks

Scalable and Decentralized Algorithms for Anomaly Detection via Learning-Based Controlled Sensing

Geethu Joseph, *Member, IEEE*, Chen Zhong, M. Cenk Gursoy, *Senior Member, IEEE*, Senem Velipasalar, *Senior Member, IEEE*, and Pramod K. Varshney, *Life Fellow, IEEE*

***Abstract*—We address the problem of sequentially selecting and observing processes from a given set to find the anomalies among them. The decision-maker observes a subset of the processes at any given time instant and obtains a noisy binary indicator of whether or not the corresponding process is anomalous. We develop an anomaly detection algorithm that chooses the processes to be observed at a given time instant, decides when to stop taking observations, and declares the decision on anomalous processes. The objective of the detection algorithm is to identify the anomalies with an accuracy exceeding the desired value while minimizing the delay in decision making. We devise a centralized algorithm where the processes are jointly selected by a common agent as well as a decentralized algorithm where the decision of whether to select a process is made independently for each process. Our algorithms rely on a Markov decision process defined using the marginal probability of each process being normal or anomalous, conditioned on the observations. We implement the detection algorithms using the deep actor-critic reinforcement learning framework. Unlike prior work on this topic that has exponential complexity in the number of processes, our algorithms have computational and memory requirements that are both polynomial in the number of processes. We demonstrate the efficacy of these algorithms using numerical experiments by comparing them with state-of-the-art methods.**

***Index Terms*—Active hypothesis testing, deep learning, reinforcement learning, actor-critic algorithm, quickest state estimation, sequential decision-making, sequential sensing.**

I. Introduction

WE CONSIDER the problem of observing a given set of processes to detect the anomalies among them via controlled sensing. Here, the decision-maker does not observe all the processes at each time instant, but sequentially selects and observes a small subset of processes at a time. The sequential control of the observation process is referred to as controlled sensing. The challenge here is to devise a selection policy to sequentially choose the processes to be observed so that the decision is accurate and fast. This problem arises, for instance, in sensor networks used for remote health monitoring, structural health monitoring, etc [1], [2]. Such systems are equipped with different types of sensors to monitor different functionalities (or processes) of the system. The sensors send their observations to a common decision-maker that identifies any potential system malfunction. These sensor observations can be noisy due to faulty hardware or unreliable communication links. Therefore, to ensure the accuracy of the decision, we employ a sequential process selection strategy that observes a subset of processes over multiple time instants before the final decision is made. Further, the different processes can be statistically dependent on each other, and as a result, observing one process also gives information about the other dependent processes. Our goal is to derive a selection policy that accurately identifies the anomalous processes with minimum delay by exploiting their underlying statistical dependence.

A. Related Literature

Anomaly detection problem is a well-studied research topic, and there are several active sensing schemes for anomaly detection designed to monitor the environments [3], [4], [5], [6]. This framework applies to various applications such as spectrum sensing in cognitive radio networks, medical diagnosis, target localization, and adaptive group testing [7]. A popular approach for solving this active sequential detection problem is to use the active hypothesis testing framework [8], [9]. Here, the decision-maker defines a hypothesis corresponding to each of the possible states of the processes and computes the posterior probabilities over the hypothesis set using the observations. The decision-maker continues to collect observations until the probability corresponding to one of the hypotheses exceeds the desired confidence level. The above active sequential detection problem can be formulated as a dynamic programming problem which is equivalent to a Markov decision process (MDP). This framework of active hypothesis testing was introduced by Chernoff in [10] which dates back to 1959. This seminal work also established the asymptotical optimality of the test for binary hypotheses. Later, the asymptotic optimality of the test was extended to the multiple hypothesis testing problem [11]. Following the Chernoff test, several other model-based tests were also studied in the literature [7], [12], [13], [14], [15]. While the detection accuracy of these algorithms is of significant research interest,

Manuscript received 22 February 2023; revised 27 June 2023 and 11 August 2023; accepted 24 August 2023. Date of publication 11 September 2023; date of current version 21 September 2023. An earlier version of this paper was presented at the IEEE International Conference on Communications, June 2021, Montreal, Canada. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tirza Routtenberg. *(Corresponding author: Geethu Joseph.)*

Geethu Joseph is with the Signal Processing Systems Group, Delft University of Technology, 2628 Delft, The Netherlands (e-mail: g.Joseph@tudelft.nl).

Chen Zhong, M. Cenk Gursoy, Senem Velipasalar, and Pramod K. Varshney are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: czhong03@syr.edu; mcgursoy@syr.edu; svelipas@syr.edu; varshney@syr.edu).

Digital Object Identifier 10.1109/TSIPN.2023.3313818

several studies focus on the sensing costs and/or switching costs during the detection process [15], [16], [17], [18], [19].

Most of the above works focus on centralized algorithms where the processes are selected jointly by a common agent. A few other works discuss the model-based non-adaptive detection algorithms in a decentralized setting [20], [21], [22], [23], [24], [25]. Recently, a model-based active hypothesis testing algorithm was also studied [26]. However, this work considered the case wherein each sensor estimated the true hypothesis independently, and the network arrived at a consensus based on the sensor decisions and associated confidence. This framework does not address the case wherein one sensor can observe only one process, and they cooperate to detect all the process states.

Recently, a new mathematical formulation of the active hypothesis testing problem based on maximizing the expectation of average reward of an MDP has been introduced in [27]. This work studies the theoretical underpinning of the framework, including the asymptotic optimality of the solution strategies, and demonstrates the superior performance of the formulation via a heuristic algorithm compared to the existing methods such as extrinsic Jensen-Shannon divergence [13] and Chernoff test-based open loop verification [7], [10], [11]. Further, for MDPs with finite state and action spaces, the average reward formulation and discounted reward formulation are equivalent for sufficiently large discount factors [28]. Based on this observation, an approximation to active sequential detection problems as the problem of discounted reward maximization of an MDP has been proposed in [29]. Following this idea, the active hypothesis testing framework has been combined with deep learning algorithms to design data-driven anomaly detection algorithms [8], [9], [29], [30]. These algorithms learn from a training dataset and provide the added advantage of adaptability to the underlying statistical dependence among the processes. The state-of-the-art algorithms employ the reinforcement learning (RL) algorithms such as Q-learning [29] and actor-critic [8], [9], [30], [31]; and the active inference framework [30]. However, since each process can either be normal or anomalous (two states per process), the number of hypotheses increases exponentially with the number of processes, leading to a heavy computational burden. Hence, we aim for a scalable anomaly detection algorithm with polynomial complexity in the number of processes.

We note that the above existing literature on deep active hypothesis testing thus far focuses on centralized algorithms where the processes are selected jointly by a common agent. As we mentioned above, centralized algorithms do not scale with the number of processes. Also, the central decision-making agent introduces a single point of failure, thus rendering this architecture is not suitable for monitoring critical applications. So to handle this problem, we also explore a decentralized algorithm for anomaly detection. In the decentralized version, there is no common agent that collects the observations and makes the process selection decisions. Instead, each sensor independently decides whether or not to observe their corresponding processes. In short, in this article, we aim to devise a learning-based controlled sensing framework for anomaly detection with polynomial complexity in the number of processes for the centralized and decentralized cases.

B. Our Contributions

The specific contributions of the article are as follows:

1) Detection Algorithm: In Section III, we first reformulate the problem of anomaly detection in terms of the marginal (not joint) probability of each process being normal or anomalous, conditioned on the observations. Consequently, the number of posterior probabilities computed by the algorithm at every time instant is linear in the number of processes. Based on these marginal posterior probabilities, we define a confidence level proportional to the decision accuracy. We then derive an algorithm in Section III for estimating the states of the processes when the confidence level exceeds a predefined threshold.

2) Centralized Algorithm: In Section IV, we present a novel centralized algorithm for anomaly detection where we restrict the number of processes chosen at any time to one. To obtain the algorithm, we define two reward functions that monotonically increase with the decision accuracy and decrease with the duration of the observation acquisition phase. These definitions allow us to reformulate the anomaly detection problem as a long-term average reward maximization problem in an MDP whose state is the marginal probability vector. The problem is solved using the deep actor-critic method.

3) Decentralized Algorithm: In Section V, we propose a decentralized version of our centralized algorithm. Here, at each time instant, the selection decision is independently made for each process, and as a consequence, the algorithm chooses more than one process at a time. The number of observations is reduced by modifying the MDP reward to accommodate the sensing cost. Using the modified notion of MDP, we derive a novel actor-critic algorithm with a new architecture and training procedure for decentralized anomaly detection.

4) Empirical Results: Using numerical results, we compare our algorithms to the state-of-the-art algorithms in Section VI. We show that the algorithms can learn and adapt to the statistical dependence among the processes and offer a good accuracy level in detecting anomalies.

Overall, this article presents centralized and decentralized algorithms for anomaly detection with polynomial complexity, which are practically more useful. Furthermore, this journal article makes several new contributions compared to the conference version [32]. In addition to the entropy-based reward function in the conference version, we also present a log-likelihood ratio (LLR)-based reward function for the scalable algorithms (see Section IV). This scheme is empirically shown to slightly outperform the entropy-based scheme. We further introduce the concept of decentralized anomaly detection and present a new actor-critic algorithm based on the concept of *centralized training and decentralized execution* (see Section V). Also, we provide a detailed derivation of recursive updates for the marginal probabilities (see Section III) and the pairwise probabilities (see Section VI).

II. Anomaly Detection Problem

We consider a set of N possibly statistically dependent processes where the state of each process is a binary random variable. The process state vector is random vector $\boldsymbol{s} \in \{0,1\}^N$ whose ith entry $\boldsymbol{s}_i$ being 0 and 1 indicates that the ith process is normal and anomalous, respectively. The statistical dependence is defined by the joint prior distribution of all the entries of $\boldsymbol{s}$, which specifies the prior probabilities associated with the 2^N possible values of $\boldsymbol{s}$. We do not assume the knowledge of this joint prior distribution although it can be learned from the training data.

Each process is monitored by a sensor. We detect the process states $\boldsymbol{s}$ by selecting and observing a subset of the processes at every time instant. Let $\mathcal{A}(k) \subseteq \{1,2,\ldots,N\}$ be the set of processes observed by the algorithm at time instant k, and the corresponding observation be $\boldsymbol{y}_{\mathcal{A}(k)}(k) \in \{0,1\}^{|\mathcal{A}(k)|}$. We assume that every observation has a fixed probability of being erroneous. Specifically, for any $i \in \{1,2,\ldots,N\}$,

$$\boldsymbol{y}_i(k) = \begin{cases} \boldsymbol{s}_i & \text{with probability } 1-p, \\ 1-\boldsymbol{s}_i & \text{with probability } p, \end{cases} \tag{1}$$

where $p \in [0,1]$ is called the flipping probability. We note that (1) provides the distribution of the measurements given the process state vector $\boldsymbol{s}$. Further, we assume that conditioned on the value of $\boldsymbol{s}$, the observations obtained across time are jointly (conditionally) independent,

$$\mathbb{P}\left[\{\boldsymbol{y}_i(l), i=1,2,\ldots,N\}_{l=1}^{k} \big| \boldsymbol{s}\right] = \prod_{i=1}^{N}\prod_{l=1}^{k} \mathbb{P}\left[\boldsymbol{y}_i(l)\big|\boldsymbol{s}_i\right], \tag{2}$$

for any k. Therefore, the observations corresponding to the ith process $\{\boldsymbol{y}_i(k) \in \{0,1\}\}_{k=1}^{\infty}$ is a sequence of independent and identically distributed binary random variables parameterized by the process state $\boldsymbol{s}_i \in \{0,1\}$. We aim to detect the anomalous processes using the observations, which is equivalent to obtaining the process state vector $\boldsymbol{s}$.

At each time instant k, the decision-maker computes an estimate of $\boldsymbol{s}$ along with the confidence in the estimate using $\boldsymbol{y}_{\mathcal{A}(k)}(k)$. Using the available information, the posterior belief $\boldsymbol{\pi}(k) \in [0,1]^{2^N}$ on process state vector $\boldsymbol{s}$ at time k is

$$\boldsymbol{\pi}(k) = \mathbb{P}\left[\boldsymbol{s} \big| \left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k}\right].$$

Since the estimate $\hat{\boldsymbol{s}}(k)$ of $\boldsymbol{s}$ at time k corresponds to $\arg\max_i \boldsymbol{\pi}_i(k)$, the detection accuracy is given by $\max_i \boldsymbol{\pi}_i(k)$. The decision-maker observes the processes until the detection accuracy exceeds the desired level denoted by $\Upsilon_{\text{upper}} \in (0,1)$, i.e., $\max \boldsymbol{\pi}(K) \geq \Upsilon_{\text{upper}}$. If multiple processes can be observed at a given time, we introduce an associated sensing cost denoted by $\lambda > 0$. Thus, at time k, the sensing cost of the network is $\lambda|\mathcal{A}(k)|$. Our goal is to optimize both sensing cost (which is proportional to the number of observations) and the duration K of the observation acquisition phase that achieves the desired level Υ_{upper}.

We look for a sequential sensor selection policy $\mathcal{A}(k)$ that adapts to the posterior belief, which, in turn, depends on the past sensor selections. To tackle this problem, we can formulate a dynamic optimization problem where the objective is to optimize for detection accuracy and sensing cost achieved in a trajectory of sensor selection $\mathcal{A}(k)$, for $k = 1,2,\ldots,$ until the accuracy $\boldsymbol{\pi}(k) \geq \Upsilon_{\text{upper}}$, i.e.,

$$\max_{\substack{\mathcal{I},K,\mathcal{A}(k),\\ k=1,2\ldots,K}} \mathbb{E}\left\{\sum_{k=1}^{K} r(k) - \eta\lambda\,|\mathcal{A}(k)|\right\}$$
$$\text{s. t. } \max \boldsymbol{\pi}(K-1) < \Upsilon_{\text{upper}}, \tag{3}$$

where $\eta > 0$ is the regularizer and $\lambda|\mathcal{A}(k)|$ is the sensing cost at time k. Also, $r(k)$ is the reward function at time k as a function of the sensor selection $\mathcal{A}(k)$ and the process state detection algorithm $\mathcal{I}$. We reward the decision-maker for choosing $\mathcal{A}(k)$ leading to observation vector $\boldsymbol{y}_{\mathcal{A}(k)}(k)$ that builds higher confidence $\max_k \boldsymbol{\pi}(k)$ in the estimated state $\hat{\boldsymbol{s}}(k)$ returned by the detection algorithm $\mathcal{I}$. A detailed discussion on the choice of the reward function is deferred to Sections IV and V.

Furthermore, we note that the constraint $\max \boldsymbol{\pi}(K-1) < \Upsilon_{\text{upper}}$ is not independent of the reward function. A higher confidence $\max_k \boldsymbol{\pi}(k)$ in the estimated process state implies higher detection accuracy. Therefore, as the decision-maker builds more confidence in the estimated process state, $\max_k \boldsymbol{\pi}(k)$ increases, and at some finite time K, it violates the constraint. After this time instant, the decision-maker can no longer accumulate the reward, and the observation acquisition phase ends. Overall, the reward formulation ensures that the stopping time K is the earliest time at which the detection accuracy exceeds the desired level Υ_{upper}. We reiterate that K is not directly included in the discounted reward, but the optimal sensor selection and process detection strategy that maximizes the reward function also minimizes the stopping time. We discuss this point again in detail in Section IV-A.

The objective function in (3) poses an additional challenge that the sum keeps growing with K. To scale down the rewards, we use a discounted reward formulation. With a suitable choice of $r(k)$, the overall underlying dynamic programming problem is

$$\max_{\substack{\mathcal{I},K,\mathcal{A}(k),\\ k=1,2\ldots,K}} \mathbb{E}\left\{\sum_{k=1}^{K} \gamma^{k-1}\left[r(k) - \eta\lambda\,|\mathcal{A}(k)|\right]\right\}$$
$$\text{s. t. } \max \boldsymbol{\pi}(K-1) < \Upsilon_{\text{upper}}. \tag{4}$$

Here, $\gamma \in (0,1)$ is the discount factor, and K is the duration of the observation acquisition phase or the time until which the decision-maker can accumulate the reward. The discounted reward formulation implies that a reward received k time steps in the future is worth only γ^k times what it would be worth if received immediately. This formulation encourages the decision-maker to build confidence in the estimate as quickly as possible.

From (4), we have two interrelated tasks: one, to develop a detection algorithm $\mathcal{I}$ to infer the process state vector $\boldsymbol{s}$ and the associated detection accuracy; and two, to design a reward function $r(k)$, sensor selection policy $\mathcal{A}(k)$, and criterion to stop collecting observations. Our goal is to develop scalable algorithms that have polynomial time complexity in N, and

we consider the centralized and decentralized settings. In the centralized setting, the algorithm selects the processes to be probed using a centralized agent, whereas in the decentralized setting, the selection decision is made independently across the processes. We next present our detection algorithm $\mathcal{I}$ that is common to both centralized and decentralized algorithms.

III. Detection Algorithm

In this section, we derive an algorithm to detect the process states from the observations. We note that the observations depend on the selection policy, and the policy design, in turn, depends on the detection algorithm. Moreover, we design a detection algorithm that minimizes the discounted reward function in (4) by building estimate confidence quickly.

To derive the detection algorithm, we exploit the underlying modeling assumptions (1) and (2) to derive a model-based detection algorithm. The optimal approach to detect the process states is to compute the joint posterior probabilities $\boldsymbol{\pi}(k)$ associated with the 2^N possible values of $\boldsymbol{s}$ using the available observations. Nonetheless, this approach is not scalable as its complexity increases exponentially with N. Thus, our detection framework relies on the marginal posterior probabilities associated with each process i.e., we use the belief vector $\boldsymbol{\sigma}(k) \in [0,1]^N$ at time k whose ith entry $\boldsymbol{\sigma}_i(k)$ is the posterior probability that the ith process is normal ($\boldsymbol{s}_i = 0$). So, the probability that the ith process is anomalous is $1 - \boldsymbol{\sigma}_i(k)$. This novel approach ensures that the computational and memory cost of the algorithm is polynomial in N.

Further, using our observation model in (1) and (2) and some algebraic simplifications, the belief vector $\boldsymbol{\sigma}_i$ can be recursively updated. The recursive update for the ith component of $\boldsymbol{\sigma}(k)$ is given by

$$\boldsymbol{\sigma}_i(k) \approx \frac{\boldsymbol{\sigma}_i(k-1)\prod_{a\in\mathcal{A}(k)} \mathbb{P}[\boldsymbol{y}_a(k)|\boldsymbol{s}_i=0]}{\sum_{s=0,1} |s-\boldsymbol{\sigma}_i(k-1)| \prod_{a\in\mathcal{A}(k)} \mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_i = s\right]}, \tag{5}$$

for $i = 1, 2, \ldots, N$. Here, the conditional probability $\mathbb{P}[\boldsymbol{y}_a(k)\big|\boldsymbol{s}_i = s]$ is given by

$$\mathbb{P}\left[\boldsymbol{y}_a(k)|\boldsymbol{s}_i = ps\right] = \sum_{s'=0,1} p^{|s'-\boldsymbol{y}_a(k)|}(1-p)^{|1-s'-\boldsymbol{y}_a(k)|} \times \mathbb{P}\left[\boldsymbol{s}_a = s'|\boldsymbol{s}_i = s\right], \tag{6}$$

which follows from (1). The term $\mathbb{P}[\boldsymbol{s}_a = s'|\boldsymbol{s}_i = s]$, which depends on the statistical dependence between the processes, can be either assumed to be known or easily estimated from the training data[1] for every pair (i, j). The approximation is exact when $\boldsymbol{s}_{\mathcal{A}(k)}$ is a deterministic function of $\boldsymbol{s}_i$. Please refer to the appendix for the detailed derivation.

Furthermore, when $\boldsymbol{s}_{\mathcal{A}(k)}$ and $\boldsymbol{s}_i$ are statistically independent, for any s and $a \in \mathcal{A}(k)$, we obtain $\mathbb{P}[\boldsymbol{y}_a(k)\big|\boldsymbol{s}_i = s] = \mathbb{P}[\boldsymbol{y}_a(k)]$. Consequently, (5) reduces to $\boldsymbol{\sigma}_i(k) = \boldsymbol{\sigma}_i(k-1)$. This update is intuitive since an observation from process $\boldsymbol{s}_{\mathcal{A}(k)}$ does not change the probabilities associated with an independent process $\boldsymbol{s}_i$. In other words, the recursive relation is exact when $\boldsymbol{s}_i$ and $\boldsymbol{s}_{\mathcal{A}(k)}$ are independent or $\boldsymbol{s}_{\mathcal{A}(k)}$ is a deterministic function of $\boldsymbol{s}_i$.

Once $\boldsymbol{\sigma}(k)$ is obtained, the estimate of the process state vector with the highest associated confidence is straightforward. The ith component of the process state vector estimate denoted by $\hat{\boldsymbol{s}}(k)$ can be determined as

$$\hat{\boldsymbol{s}}_i(k) = \begin{cases} 0 & \text{if } \boldsymbol{\sigma}_i(k) \geq 1 - \boldsymbol{\sigma}_i(k) \\ 1 & \text{if } \boldsymbol{\sigma}_i(k) < 1 - \boldsymbol{\sigma}_i(k). \end{cases} \tag{7}$$

We note that the detection accuracy can be related to the marginal posterior probabilities via the Fréchet and max-min inequalities as follows.

$$\begin{aligned} \max_j \boldsymbol{\pi}_j(k) &= \max_{\tilde{\boldsymbol{s}}} \mathbb{P}\left[\boldsymbol{s} = \tilde{\boldsymbol{s}}|\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^k\right] \\ &\leq \max_{\tilde{\boldsymbol{s}}} \min_{i=1,2,\ldots,N} \mathbb{P}\left[\boldsymbol{s}_i = \tilde{\boldsymbol{s}}_i|\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^k\right] \\ &\leq \min_{i=1,2,\ldots,N} \max_{\tilde{\boldsymbol{s}}_i=0,1} \mathbb{P}\left[\boldsymbol{s}_i = \tilde{\boldsymbol{s}}_i|\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^k\right] \\ &= \min_{i=1,2,\ldots,N} \max\{\boldsymbol{\sigma}_i(k), 1-\boldsymbol{\sigma}_i(k)\}, \end{aligned} \tag{8}$$

where $\max\{\boldsymbol{\sigma}_i(k), 1-\boldsymbol{\sigma}_i(k)\}$ is the detection accuracy for the ith process. We replace the constraint $\max_i \boldsymbol{\pi}_i(K-1) < \Upsilon_{\text{upper}}$ of the optimization problem (4) as

$$\min_{i=1,2,\ldots,N} \max\{\boldsymbol{\sigma}_i(k-1), 1-\boldsymbol{\sigma}_i(K-1)\} < \Upsilon_{\text{upper}}. \tag{9}$$

The above condition ensures that the detection accuracy associated with at least one of the processes does not exceed the desired level Υ_{upper} at time $K-1$. Consequently, the constraint forces the decision-maker to continue the observation acquisition phase until the detection accuracy associated with all the processes exceeds the desired level to maximize the objective function.

Hence, the derivation of the detection algorithm is complete and we next discuss the design of the selection policy. The design of the selection policy (i.e., the policy to determine which processes to observe at a given time) is a sequential decision-making problem, and this problem can be formulated using the mathematical framework of MDPs. The MDP-based formulation allows us to obtain the selection policy via reward maximization of the MDP using RL algorithms. In the following sections, we present the MDP and RL algorithms for the centralized and decentralized settings.

IV. Centralized Decision-Making

In the centralized version, we restrict $|\mathcal{A}(k)| = 1$ for all values of k for simplicity, i.e., only one process is observed per time instant. The observation obtained by a sensor at a given time is sent to a centralized decision-making agent that utilizes an MDP framework to decide on which process to observe next. The MDP-based selection policy is as follows.

A. Markov Decision Process

In the centralized case, the MDP is defined as follows:

1) MDP State: Our detection algorithm is based on the belief vector $\boldsymbol{\sigma}(k)$ that changes with time after each observation

[1]Training assumes the knowledge of $\boldsymbol{s}$ but not the optimal sensor selection.

arrives. Therefore, we define $\boldsymbol{\sigma}(k) \in [0,1]^N$ as the state of the MDP at time k. We note that the MDP state vector $\boldsymbol{\sigma}(k)$ is different from the process state vector $\boldsymbol{s}$.

2) MDP Action: The state of MDP depends on the observation which in turn depends on the process selected by the policy. Naturally, the action taken by the decision maker at time instant k is the selected process $\mathcal{A}(k) \in \{1, 2, \ldots, N\}$.

3) MDP State Transition: For our problem, the MDP state $\boldsymbol{\sigma}(k)$ at time k is a deterministic function of the previous MDP state $\boldsymbol{\sigma}(k-1)$, the action $\mathcal{A}(k)$, and the observation $\boldsymbol{y}_{\mathcal{A}(k)}(k)$. So, the MDP state transition is modeled by (5) and (6).

4) Reward Function: We recall from (4) that the decision-maker is rewarded for choosing $\mathcal{A}(k)$ such that $\boldsymbol{y}_{\mathcal{A}(k)}(k)$ builds higher confidence in the estimated state $\hat{\boldsymbol{s}}(k)$. The confidence can be defined using the uncertainty associated with the ith process, which can be quantified using the entropy of its posterior distribution $[\boldsymbol{\sigma}_i(k) \;\; 1-\boldsymbol{\sigma}_i(k)]$. As a consequence, the instantaneous reward of the MDP is

$$r_{\text{entropy}}(k) = \sum_{i=1}^{N} H(\boldsymbol{\sigma}_i(k-1)) - H(\boldsymbol{\sigma}_i(k)), \tag{10}$$

where $H(x) = -x \log x - (1-x)\log(1-x)$ is the binary entropy function. Alternatively, we can also use the LLR of the posterior distribution of the processes to capture the confidence. The LLR-based instantaneous reward function is

$$r_{\text{LLR}}(k) = \sum_{i=1}^{N} L(\boldsymbol{\sigma}_i(k)) - L(\boldsymbol{\sigma}_i(k-1)), \tag{11}$$

where $L(x) = x \log(x/(1-x)) + (1-x)\log((1-x)/x)$. The differences of the entropies and LLRs quantify the reduction in uncertainty from time $k-1$ to k. Hence, these reward formulations force the RL agent to reduce the uncertainty and detect with higher confidence as quickly as possible.

We make a few observations about the reward function before we present the process selection policy. As we mentioned in Section II, from (8), (10), and (11), both reward and detection accuracy are functions of $\boldsymbol{\sigma}(k)$, which in turn, depends on sensor selection $\mathcal{A}(k)$ and detection algorithm $\mathcal{I}$. For any given $\boldsymbol{\sigma}(k-1)$, the entropy-based reward function satisfies

$$\underset{\boldsymbol{\sigma}(k)\in[0,1]^N}{\arg\max}\; r_{\text{entropy}}(k) = \underset{\boldsymbol{\sigma}\in[0,1]^N}{\arg\min} \sum_{i=1}^{N} H(\boldsymbol{\sigma}_i) = \{0,1\}^N,$$

where $\{0,1\}^N$ is the set of 2^N vectors whose entries are either 0 or 1. Similarly, it is easy to show that

$$\underset{\boldsymbol{\sigma}(k)\in[0,1]^N}{\arg\max}\; r_{\text{entropy}}(k) = \frac{1}{2}\mathbf{1},$$

where $\mathbf{1}$ is the all-ones vector. This result implies that the reward function encourages the decision-maker to move away from the non-informative posterior and towards an informative posterior $\boldsymbol{\sigma}(k) \in \{0,1\}^N$. Moreover, from (8), as $\boldsymbol{\sigma}$ moves towards a vector in the set $\{0,1\}^N$, the detection accuracy increases. In other words, the reward function drives the algorithm to build the confidence in detection and improve detection accuracy. Similar results and observations hold for the log-likelihood-based reward function. This observation agrees with the discussion in Section II that as the decision-maker builds more confidence in the estimate, the detection accuracy increases, and the observation acquisition phase ends at a finite time K.

Since both reward functions behave similarly, in the sequel, we use $r(k)$ to denote the reward function at time k, which is either $r_{\text{entropy}}(k)$ or $r_{\text{LLR}}(k)$. Further, we ignore the sensing cost as $\lambda|\mathcal{A}(k)| = \lambda$ is fixed and independent of the sensor selection $\mathcal{A}(k)$. Then, dynamically solving (4) at time k is the same as maximizing the long term discounted reward of the MDP $\bar{R}(k) = \sum_{l=k}^{K} \gamma^{l-k} r(l)$, where γ is defined in (4). We next use the actor-critic algorithm to solve the long-term average reward maximization problem.

B. Actor-Critic Algorithm

We use the actor-critic algorithm which is a deep learning-based RL technique to generate a sequential policy that maximizes the long-term expected discounted reward $\bar{R}(k)$ of a given MDP. Unlike the detection algorithm given in Section III, here, the optimal solution for policy design can not be derived in closed form. Therefore, we resort to the deep learning framework which learns the optimal sensor selection from the training data, which is known to have a superior performance over the model-based approximation algorithms [29], [30].

The actor-critic framework maximizes the discounted reward using two neural networks: actor and critic networks. The actor learns a stochastic policy that maps the state of the MDP to a probability vector on the set of actions. The critic learns a function that evaluates the policy of the actor and gives feedback to the actor. As a result, the two neural networks interact and adapt to each other. The components of the actor-critic algorithm are as follows:

1) Actor Network: The actor takes the state of the MDP $\boldsymbol{\sigma}(k-1) \in [0,1]^N$ as its input. Its output is the probability vector $\boldsymbol{\mu}(\boldsymbol{\sigma}(k-1); \alpha_{\text{cen}}) \in [0,1]^N$ over the set of processes where α_{cen} denotes the set of parameters of the actor network. The decision maker employs a stochastic process selection strategy $\mathcal{A}(k) \sim \boldsymbol{\mu}(\boldsymbol{\sigma}(k-1); \alpha_{\text{cen}})$, i.e., the ith process is selected at time k with probability equal to the ith entry $\boldsymbol{\mu}_i(\boldsymbol{\sigma}(k-1); \alpha_{\text{cen}})$ of the actor output.

2) Reward Computation: Once the process $\mathcal{A}(k)$ is selected, the decision maker receives the corresponding observation $\boldsymbol{y}_{\mathcal{A}(k)}$, and the MDP state $\boldsymbol{\sigma}(k-1)$ is updated to $\boldsymbol{\sigma}(k)$ as given by (5). The decision maker also calculates the instantaneous reward $r(k)$ using (10) or (11), and the reward value is utilized by the critic network to provide feedback to the actor network, as discussed next.

3) Critic Network: The critic neural network models the value function $V(\boldsymbol{\sigma}(k))$ of the current MDP state as

$$V^{\mu}(\boldsymbol{\sigma}) = \mathbb{E}_{\mathcal{A}(k)\sim\mu}\left\{\bar{R}(k) \middle| \boldsymbol{\sigma}(k) = \boldsymbol{\sigma}\right\}.$$

Here, $V^{\mu}(\boldsymbol{\sigma})$ is the expected average future reward when the MDP starts at state $\boldsymbol{\sigma}$ and follows the policy $\mu(\cdot;\theta)$ thereafter, indicating the long-term desirability of the MDP state $\boldsymbol{\sigma}$. Consequently, the input to the critic network is the posterior vector $\boldsymbol{\sigma} \in [0,1]^N$, and the output is the learned value function

$\hat{V}(\boldsymbol{\sigma};\beta_{\text{cen}})$. Here, β_{cen} is the set of parameters of the critic neural network. The scalar critique for the actor network is the temporal difference (TD) error $\delta(k;\beta_{\text{cen}})$ defined as

$$\delta(k;\beta_{\text{cen}}) = r(k) + \gamma\hat{V}(\boldsymbol{\sigma}(k)) - \hat{V}(\boldsymbol{\sigma}(k-1)). \tag{12}$$

A positive TD error indicates that the probability of choosing the current action should be increased in the future, and a negative TD error suggests to decrease the probability of $\mathcal{A}(k)$.

4) Learning Actor Parameters: The goal of the actor is to choose a policy such that the value function is maximized which, in turn, maximizes the expected average future reward. Therefore, the actor updates its parameter set α_{cen} using the gradient descent step by moving in the direction in which the value function is maximized, i.e.,

$$\alpha_{\text{cen}} = \alpha_{\text{cen}}^{-} + \delta(k;\beta_{\text{cen}})\nabla_{\alpha_{\text{cen}}}[\log \mu_{\mathcal{A}(k)}(\boldsymbol{\sigma}(k-1);\alpha_{\text{cen}})], \tag{13}$$

where α_{cen}^{-} is the previous network estimate [33, Ch.13].

5) Learning Critic Parameters: The critic chooses its parameters such that it learns the estimate $\hat{V}$ of the state value function V accurately. So, the critic updates its parameter set β_{cen} by minimizing the squared TD error $\delta^2(k;\beta_{\text{cen}})$.

6) Termination Criterion: The actor-critic algorithm continues to collect observations until the constraint in (4) is satisfied because afterwards, the decision maker can not accumulate the reward. From (8), the constraint violation in (4) reduces to

$$\min_{i=1,2,\ldots,N} \max\{\boldsymbol{\sigma}_i(k), 1-\boldsymbol{\sigma}_i(k)\} > \Upsilon_{\text{upper}}. \tag{14}$$

Once the stopping time K satisfying (14) is reached, no further observations are acquired. Thus, the actor-critic algorithm is completed, and we next summarize the overall algorithm.

Algorithm 1: Centralized Actor-Critic RL for Anomaly Detection: Training Phase

Parameters: Discount rate $0<\gamma<1$, time steps per episode T
Initialization: Initialize α_{cen}, β_{cen} randomly, $\boldsymbol{\sigma}(0)$
for Episode index $= 1, 2, \ldots$ **do**
  Time index $k = 1$
  repeat
    Choose a process $\mathcal{A}(k) \sim \boldsymbol{\mu}(\boldsymbol{\sigma}(k-1), \alpha_{\text{cen}})$
    Receive observation $\boldsymbol{y}_{\mathcal{A}(k)}(k)$
    Compute $\boldsymbol{\sigma}(k)$ using (5) and (6), and instantaneous reward $r(k)$ using (10) or (11)
    Update α_{cen} using (13), and β_{cen} as the minimizer of $\delta^2(k;\beta_{\text{cen}})$ in (12)
    Increase time index $k = k+1$
  until $k > T$
end for

Algorithm 2: Centralized Actor-Critic RL for Anomaly Detection: Testing Phase.

Parameters: Upper threshold on confidence Υ_{upper}
Initialization: α_{cen} and $\boldsymbol{\sigma}(0)$ from the training phase
Time index $k = 1$
repeat
  Choose a process $\mathcal{A}(k) \sim \boldsymbol{\mu}(\boldsymbol{\sigma}(k-1), \alpha_{\text{cen}})$
  Receive observation $\boldsymbol{y}_{\mathcal{A}(k)}(k)$
  Compute $\boldsymbol{\sigma}(k)$ using (5) and (6)
  Increase time index $k = k+1$
until (14) is satisfied
Declare the estimate $\hat{\boldsymbol{s}}$ using (7)

C. Overall Algorithm

Combining the detection algorithm in Section III and the deep actor-critic method in Section IV-B, we obtain our centralized anomaly detection algorithm. The decision-maker collects observations using the selection policy obtained using the actor-critic algorithm until the stopping criterion given in (14) is satisfied. After the actor-critic algorithm terminates, the decision-maker computes $\hat{\boldsymbol{s}}$ using (7). We present the pseudo-code of the overall procedure in Algorithms 1 and 2. Our algorithm balances the model-based and deep learning-based approaches via a model-based detection algorithm and neural network-based actor-critic method, resulting in a model-assisted learning algorithm. The model-based part helps to reduce the training complexity of the deep neural networks, requiring them to learn only the selection policy that can not be optimally derived using the model-based approach.

The computational complexity of our algorithm is determined by the neural networks' size, the update of the posterior belief vector given by (5) and (6), and the reward computation in (10) or (11). Since all of them have linear complexity in the number of processes N, the overall computational complexity of our algorithm is polynomial in N. Also, the sizes of all the variables involved in the algorithm are linear in N except for the pairwise conditional probability $\mathbb{P}[\boldsymbol{s}_i|\boldsymbol{s}_j]$ for $i, j = 1, 2, \ldots, N$. Therefore, the memory requirement of the algorithm is $\mathcal{O}(N^2)$. Hence, our algorithm possesses polynomial complexity, unlike the anomaly detection algorithms in [9], [30] with exponential complexity.

It is straightforward to extend our algorithm to the case in which the decision maker chooses at most $1 \leq n \leq N$ processes at a time. In that case, the state remains unchanged, but the action space of the MDP changes. When we allow up to n processes to be observed, there are $\sum_{l=1}^{n}\binom{N}{l}$ possible actions. So, the output layer of the actor has $\sum_{l=1}^{n}\binom{N}{l}$ neurons, each representing one action. Therefore, the overall computational complexity of the resulting algorithm is polynomial in N^n, and the memory requirement is $\mathcal{O}(N^n)$. Naturally, as n grows, the centralized algorithm becomes non-scalable.

V. Decentralized Decision Making

In the decentralized setting, there is no centralized decision-making agent that accumulates the observations and makes the process selection decisions. At every time instant, the sensors independently decide whether or not to observe their corresponding processes. The sensors that choose to observe the corresponding processes collect the observations. Further, depending on the underlying network topology, the sensors share

their observations. In particular, the ith sensor (i.e., the sensor corresponding to the ith process) can receive observations from a set of neighboring sensors denoted by $\mathcal{N}_i \subseteq \{1, 2, \ldots, N\}$, including the ith sensor itself. In other words, at time k, the ith sensor knows the observations corresponding to the processes indexed by $\mathcal{N}_i \cap \mathcal{A}(k)$. Similarly, if the ith sensor observes the corresponding process at time k (i.e., $i \in \mathcal{A}(k)$), the observation $\boldsymbol{y}_i(k)$ is also available at the sensors indexed by the set $\{j : i \in \mathcal{N}_j\}$.

Each sensor keeps its local estimate of the marginal posterior probabilities. We denote the posterior probability vector of the ith sensor at time k by $\boldsymbol{\sigma}^{(i)}(k)$ which is updated using $\boldsymbol{y}_{\mathcal{N}_i \cap \mathcal{A}(k)}(k)$ via (5) and (6). Further, since the sensors have potentially different posterior probability vectors, we can not directly use the stopping condition (14). So, the ith sensor broadcasts a message when the following condition is met,

$$\max\{\boldsymbol{\sigma}_i^{(i)}(k), 1 - \boldsymbol{\sigma}_i^{(i)}(k)\} > \Upsilon_{\text{upper}}. \tag{15}$$

The sensors do not immediately stop collecting observations when (15) is satisfied. Instead, they continue to collect observations until they receive similar broadcast messages from all the sensors. When the observation acquisition phase ends at time K, each sensor i declares the estimate of its process as

$$\hat{\boldsymbol{s}}_i = \begin{cases} 0 & \text{if } \boldsymbol{\sigma}_i^{(i)}(K) \geq 1 - \boldsymbol{\sigma}_i^{(i)}(K) \\ 1 & \text{if } \boldsymbol{\sigma}_i^{(i)}(K) < 1 - \boldsymbol{\sigma}_i^{(i)}(K). \end{cases} \tag{16}$$

In short, in the decentralized version, the process selection algorithm runs at each sensor independently of each other. We next describe how the sensors decide whether or not to choose to observe the corresponding process. Similar to the centralized algorithm discussed in Section IV, we use a deep actor critic algorithm as described next.

A. Decentralized Deep Actor Critic Framework and MDP

In the decentralized algorithm, each sensor has to learn its selection policy depending on its posterior vector $\boldsymbol{\sigma}^{(i)}(k)$. Our framework consists of one actor network per sensor and a single common critic network, and we adopt the mechanism of *centralized training and decentralized execution* to learn the neural networks. Specifically, we assume that all the actor networks share the same parameters and are trained together in a centralized fashion. In the testing phase, each sensor uses a separate actor network derived from the common actor network learned via centralized training. The centralized training phase assumes that all the sensors receive observations from all the other sensors. Consequently, all the sensors have the same set of observations given by $\boldsymbol{y}_{\mathcal{A}(k)}(k)$, and they share a common posterior probability vector $\boldsymbol{\sigma}(k)$. This assumption simplifies our training and leads to a common MDP in the centralized training phase as described next.

1) MDP State and Action: For the centralized training phase, the MDP state and state transitions are identical to those in the centralized algorithm, i.e., we define $\boldsymbol{\sigma}(k) \in [0, 1]^N$ as the state of the MDP at time k, and the MDP state transitions are modeled by (5) and (6). Based on the MDP state, each sensor decides whether or not to sense the corresponding process. The indices of the selected process at time k denoted by $\mathcal{A}(k) \subseteq \{1, 2, \ldots, N\}$ represent the (joint) decisions taken by each sensor and form the MDP action.

2) MDP Reward: Unlike the centralized case, here $|\mathcal{A}(k)|$ is not necessarily one. Here, all the sensors aim to achieve the common goal of minimum stopping time with a small sensing cost and the desired detection accuracy (decided by Υ_{upper}). Therefore, we need a single reward function that promotes the common goal of the network, defined as

$$r_{\text{de}}(k) = r(k) - \eta\lambda\,|\mathcal{A}(k)|\,, \tag{17}$$

where $\eta > 0$ is the regularizer and $\lambda|\mathcal{A}(k)|$ is the sensing cost of the network at time k. Also, we recall that $r(k)$ is either $r_{\text{entropy}}(k)$ in (10) or $r_{\text{LLR}}(k)$ in (11). The decentralized reward $r_{\text{de}}(k)$ encourages the sensor network to minimize the stopping time via the first term and minimize the sensing cost via the second term. In other words, λ balances the trade-off between the stopping time and sensing cost.

Using the above notion of MDP, we learn the common actor and critic networks in the centralized training phase. The critic learns a function that evaluates the policy followed by the common actor and gives a common feedback to them for the joint action $\mathcal{A}(k)$ of the network. The neural networks interact and adapt to each other during the centralized training phase. In the testing phase, the sensors choose the actions based on the actor network's learned policy without the critic network.

Before presenting the details of the testing and training phases, we compare the MDP formulations in the centralized and decentralized settings. In the centralized setting presented in Section IV, the MDP captures the common posterior evolution, and the control and reward of a single decision-maker. For the decentralized case introduced and analyzed in this section, the MDP captures the evolution of the common posterior vector, the joint actions that a group of decision-makers takes, and the resulting collective reward.

Next, we discuss the centralized training and decentralized execution phases of the decentralized decision-making.

B. Centralized Training

For decentralized actor-critic networks, the reward computation and critic network definition and learning are the same as those in the centralized algorithm (see Section IV-B). Also, the common actor network input is the MDP state $\boldsymbol{\sigma}(k-1) \in [0, 1]^N$. However, unlike the centralized algorithm, the actor network output is not a probability vector. Rather, it is $\boldsymbol{\nu}(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}}) \in [0, 1]^N$ whose ith entry is the probability of probing the ith process at time k.

In the centralized training phase, we learn the parameters of the common actor network α_{de} and the critic network, denoted by β_{de}. The learning in the centralized training is identical to that of the centralized algorithm with a few differences in learning the actor network. We recall that the actor network updates its parameters α_{de} using the gradient descent step by moving in the direction in which the value function is maximized using the probability of the action $\mathcal{A}(k)$. Here, we explicitly calculate this

Algorithm 3: Decentralized Actor-Critic RL for Anomaly Detection: (Centralized) Training phase.

Parameters: Discount rate $0<\gamma<1$, time steps per episode T
Initialization: Initialize $\alpha_{\text{de}}, \beta_{\text{de}}$ randomly, $\boldsymbol{\sigma}(0)$
for Episode index $= 1, 2, \ldots$ **do**
  Time index $k = 1$
  repeat
    $\mathcal{A}(k) = \emptyset$
    for process (or sensor) index $a = 1, 2, \ldots, N$ **do**
      Choose process $a \sim \boldsymbol{\nu}_a(\boldsymbol{\sigma}(k-1), \alpha_{\text{de}})$
      if process a is selected **then**
        Add a to $\mathcal{A}(k)$ and receive observation $\boldsymbol{y}_a(k)$
      end if
    end for
    Compute $\boldsymbol{\sigma}(k)$ using (5) and (6), and instantaneous reward $r(k)$ using (17)
    Update α_{cen} using (18), and β_{cen} as the minimizer of $\delta^2(k; \beta_{\text{de}})$ in (12)
    Increase time index $k = k + 1$
  until $k > T$
end for

Algorithm 4: Decentralized Local Actor-Critic RL for Anomaly Detection: Testing Phase at the ith sensor.

Parameters: Upper threshold on confidence Υ_{upper}
Initialization: α_{de} and $\boldsymbol{\sigma}^{(i)}(0)$ from the training phase
Time index $k = 1$
repeat
  Observe process i with prob. $\boldsymbol{\nu}_i(\boldsymbol{\sigma}^{(i)}(k-1), \alpha_{\text{de}})$
  if process i is selected **then**
    Receive $\boldsymbol{y}_i(k)$ and send it to sensors $\{j : i \in \mathcal{N}_j\}$
  end if
  Compute $\boldsymbol{\sigma}(k)$ using $\boldsymbol{y}_{\mathcal{N}_i \cap \mathcal{A}(k)}(k)$ via (5) and (6)
  if (15) is satisfied **then**
    Broadcast a message to stop
  end if
  Increase time index $k = k + 1$
until all the sensors request to stop at time k
Declare the estimate $\hat{\boldsymbol{s}}_i$ using (16)

probability as

$$\phi(\mathcal{A}(k); \alpha_{\text{de}}) = \prod_{a \in \mathcal{A}(k)} \boldsymbol{\nu}_a(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}}) \times \prod_{a \notin \mathcal{A}(k)} \left(1 - \boldsymbol{\nu}_a(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}})\right),$$

where $\nu_a(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}})$ is the probability of choosing process a at time k, and we use the fact that each process is selected independently. Thus, the actor parameters update is

$$\alpha_{\text{de}} = \alpha_{\text{de}}^{-} + \delta(k; \beta_{\text{de}}) \nabla_{\alpha_{\text{de}}} \phi(\mathcal{A}(k); \alpha_{\text{de}}), \tag{18}$$

where α_{de}^{-} is the previous network estimate. The critic updates its parameter set β_{de} by minimizing the squared TD error δ^2. The overall procedure in summarized in Algorithms 3.

C. Decentralized Execution

For the decentralized execution during the testing phase, the actor network of the ith sensor represents a stochastic policy that maps the posterior vector $\boldsymbol{\sigma}^{(i)}(k) \in [0,1]^N$ to a probability $\boldsymbol{\nu}^{(i)}(\boldsymbol{\sigma}^{(i)}(k-1) \in [0,1]$ with which the corresponding process is selected. These actor networks are identical to the common actor except for the output layer. The reduced actor network of the ith sensor retains only the ith entry of the output of the common actor network, and the remaining output nodes are removed. Thus, the output of the ith actor network is $\boldsymbol{\nu}^{(i)}(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}}) = \boldsymbol{\nu}_i(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}})$ which is the probability of choosing the ith process at time k.

At every time instant k, the ith sensor feeds its posterior vector $\boldsymbol{\sigma}^{(i)}(k-1)$ as the input to its actor network and choose to observe the corresponding process with probability equal to its actor output. Depending upon the network topology (decided by $\{\mathcal{N}_i\}_{i=1}^N$), the sensors share their observations and update the posterior vector using (5) and (6). The sensors continue to collect observations until all the sensors satisfy (15). Thus, the overall stopping criterion of the algorithm is

$$\min_{i=1,2,\ldots,N} \max\{\boldsymbol{\sigma}_i^{(i)}(k), 1 - \boldsymbol{\sigma}_i^{(i)}(k)\} > \Upsilon_{\text{upper}},$$

which is similar to (14). After the actor-critic algorithm terminates, the decision-makers compute $\hat{\boldsymbol{s}}$ using (7). We present the pseudo-code of the overall procedure in Algorithms 4.

To summarize, centralized training refers to using common actor and critic networks during training. Decentralized execution refers to independent decision-making using individual actor networks derived from the common actor network.

As a final remark on the decentralized version, we mention its relation with the centralized detection algorithms. For this, we consider the special case when all the sensors broadcast their observations, i.e., $\mathcal{N}_i = \{1, 2, \ldots, N\}$ for all i. Then, the posterior vectors at all the sensors are identical, $\boldsymbol{\sigma}^{(i)}(k) = \boldsymbol{\sigma}(k)$, for all values of $i, k > 0$. Mathematically, this system is equivalent to a centralized anomaly detection algorithm that uses the common actor learned in the centralized training phase. This centralized algorithm chooses to observe the ith process with probability given by the corresponding entry of the actor output $\boldsymbol{\nu}_i(\boldsymbol{\sigma}(k-1); \alpha_{\text{de}})$. Nonetheless, we note that this algorithm (which can potentially have $\mathcal{A}(k) > 1$) is different from the centralized algorithm discussed in Section IV (which restricts $\mathcal{A}(k) = 1$). Moreover, we note that our decentralized framework can address a spectrum of scenarios ranging from complete information exchange (as discussed above) to no information exchange (i.e., $\mathcal{N}_i = \{i\}$ for all i). Indeed, these two cases (referred to as shared and local detection algorithms) are empirically analyzed in Section VI-B.

VI. Simulation Results

This section empirically studies the detection performance of our algorithm. We use three metrics for the performance evaluation: accuracy (the fraction of times the algorithm correctly identifies all the anomalies), stopping time (K), and number of observations per unit time ($= \frac{1}{K}\sum_{k=1}^{K} |\mathcal{A}(k)|$).

Our simulation setup considers five processes $N = 5$ and assume that the probability of each process being normal is $q = 0.8$. Here, the first and second processes ($\boldsymbol{s}_1$ and $\boldsymbol{s}_2$) are statistically dependent, and the fourth and fifth processes ($\boldsymbol{s}_4$ and $\boldsymbol{s}_5$) are also statistically dependent. These pairs of processes are independent of each other and independent of the third process ($\boldsymbol{s}_3$). The dependence is captured using the correlation coefficient $\rho \in [0, 1]$ common to both process pairs:

$$\mathbb{P}[\boldsymbol{s}_1 = \boldsymbol{s}_2 = 0] = \mathbb{P}[\boldsymbol{s}_4 = \boldsymbol{s}_5 = 0] = q^2 + \rho q(1-q)$$
$$\mathbb{P}[\boldsymbol{s}_1 = \boldsymbol{s}_2 = 1] = \mathbb{P}[\boldsymbol{s}_4 = \boldsymbol{s}_5 = 1] = (1-q)^2 + \rho q(1-q)$$
$$\mathbb{P}[\boldsymbol{s}_1 \neq \boldsymbol{s}_2] = \mathbb{P}[\boldsymbol{s}_4 \neq \boldsymbol{s}_5] = (1-\rho)q(1-q).$$

The flipping probability is $p = 0.2$. Our data model is inspired by the anomaly detection framework in various applications, such as spectrum sensing in cognitive radio networks, remote sensing, remote health monitoring, and structural health monitoring, that are threshold-based, i.e., the difference between the observed data and the expected data with no anomaly is thresholded to decide whether there is an anomaly (a binary decision).

Before we present the simulation results, we note that for the above setting, the pairwise probabilities required for the posterior updates in (6) can be computed as follows. Here, there are three independent groups of process $(\{1, 2\}, \{3\}, \{4, 5\})$. If i and j are independent processes, for any $s, s' \in \{0, 1\}$, we have

$$\mathbb{P}\left[\boldsymbol{s}_i = s' \middle| \boldsymbol{s}_j = s\right] = \mathbb{P}[\boldsymbol{s}_i = s'] = \begin{cases} q & \text{if } s' = 0 \\ 1-q & \text{if } s' = 1. \end{cases}$$

Similarly, if i and j are dependent, the pairwise probabilities are given by

$$\mathbb{P}\left[\boldsymbol{s}_i = s' \middle| \boldsymbol{s}_j = s\right] = \frac{\mathbb{P}[\boldsymbol{s}_i = s', \boldsymbol{s}_i = s]}{\mathbb{P}[\boldsymbol{s}_i = s]}$$
$$= \begin{cases} q + \rho(1-q) & s' = s = 0 \\ (1-q) + \rho q & s' = s = 1 \\ (1-\rho)q & s' = 0, s = 1 \\ (1-\rho)(1-q) & s' = 1, s = 0. \end{cases}$$

Also, the prior $\boldsymbol{\sigma}(0) = q\mathbf{1}$.

In the following, we present the numerical results for the centralized and decentralized algorithms. Our centralized algorithm chooses only one process per unit time whereas our decentralized algorithm can potentially choose multiple processes per unit time. So, there is no direct comparison between them. For a fair comparison, we compare our algorithms with the competing algorithms with similar policies.

A. Centralized Algorithm

The architecture and parameters of our algorithm are as follows. We implement the actor and critic neural networks with three layers and the ReLU activation function between consecutive layers. The actor output is normalized to be a probability vector over the set of processes. The parameters of the neural networks are updated using Adam Optimizer, and we set the learning rates of the actor and the critic as 5×10^{-4}, and 5×10^{-3}, respectively. Also, the discount factor γ is 0.9.

TABLE I
COMPARISON OF (TESTING) RUNTIME FOR DIFFERENT SCHEMES

Reward function	Naive marginal	Our marginal	Joint
Entropy	0.46 ms	0.49 ms	0.68 ms
LLR	0.48 ms	0.51 ms	0.69 ms

We compare our algorithm (labeled as `Marginal`) with three other schemes that also choose one process per unit time.

1) Joint Probability Mass Function (pmf)-Based Scheme (Labeled as `Joint`*):* The state-of-the-art algorithm presented in [9] is based on the joint posterior probabilities of all the entries of process state $\boldsymbol{s}$. Since $\boldsymbol{s}$ can take 2^N possible values, its complexity is exponential in N. However, the joint probabilities help to exploit the complete statistical dependencies.

2) Naive Marginal Pmf-Based Scheme (Labeled as `Naive`*):* The naive method relies on the marginal posterior $\boldsymbol{\sigma} \in [0, 1]^N$. It is identical to our algorithm except that it only updates the entry $\boldsymbol{\sigma}_{\mathcal{A}(k)}(k)$ at time k, ignoring the possible statistical dependence among processes. Hence, the computational complexity of this algorithm is also $\mathcal{O}(N)$. We note that, unlike our algorithm, this algorithm does not use any approximation.

3) Chernoff Test (Labeled as `Chernoff`*):* This model-based test sequentially chooses actions that quickly build the posterior belief on $\boldsymbol{s}$ [10]. Its stochastic policy, given by the probabilities over all the actions $\boldsymbol{q} \in [0, 1]^N$, is

$$\underset{\substack{\boldsymbol{q}\in[0,1]^N \\ \sum_i \boldsymbol{q}_i = 1}}{\arg\max} \min_{\substack{\hat{\boldsymbol{s}}\in\{0,1\}^N \\ \hat{\boldsymbol{s}}\neq\bar{\boldsymbol{s}}(k)}} \sum_{i=1}^{N} \boldsymbol{q}_i \mathcal{D}\{p(y_i(k)|\boldsymbol{s}=\bar{\boldsymbol{s}}(k)) \,\|p(y_i(k)|\boldsymbol{s}=\hat{\boldsymbol{s}})\},$$

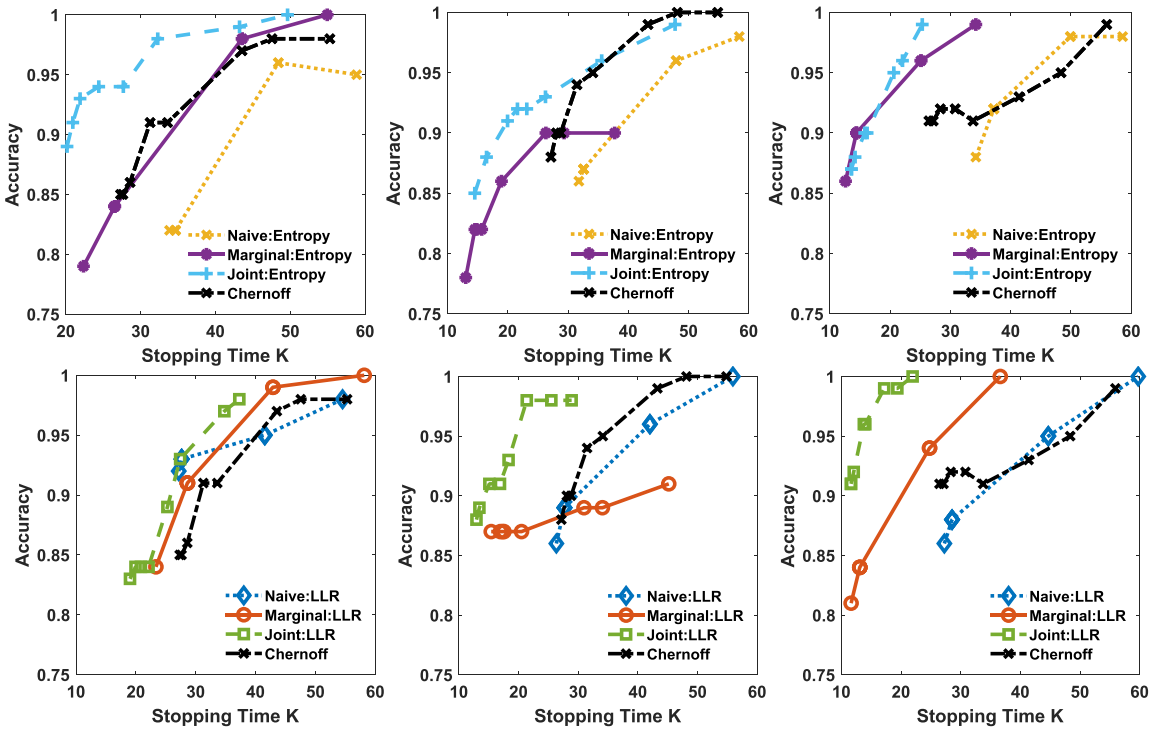
where $\mathcal{D}$ denotes the KL divergence and $\bar{\boldsymbol{s}}(k)$ is the state vector corresponding to the maximum joint posterior probability at time $k-1$. However, from (2), $p(y_i(k)|\boldsymbol{s}) = p(y_i(k)|\boldsymbol{s}_i)$. So, the KL divergence is $(1-2p)\log(1/p-1)$ if $\hat{\boldsymbol{s}}_i \neq \bar{\boldsymbol{s}}_i(k)$, and it is zero otherwise. Hence, the above optimization problem reduces to

$$\underset{\boldsymbol{q}\in[0,1]^N:\, \sum_i \boldsymbol{q}_i = 1}{\arg\max} \left(\min_i \boldsymbol{q}_i\right),$$

and the Chernoff test selects a process uniformly at random. This approach is suboptimal because the statistical dependence is accounted only for the joint posterior computation but ignored for the process selection.

We implement the deep learning algorithms using the entropy and LLR-based rewards (see (10) and (11) for the marginal pmf-based algorithms and [9] for the joint pmf-based algorithm). Since the number of observations per unit time is always one, the performance metrics are accuracy and stopping time. Our results are summarized in Fig. 1, 3, and 4 and Table I, and the key inferences are below.

Fig. 1 compares the entropy-based schemes (in the first row) and the LLR-based schemes (in the second row) whereas Fig. 3 plots the accuracy (in the first row) and the stopping time (in the second row) of LLR-based-scheme as a function of the upper threshold Υ_{upper} for confidence. The interested readers can refer



(a) $\rho = 0$ (b) $\rho = 0.8$ (c) $\rho = 1$

Fig. 1. Accuracy of the centralized anomaly detection algorithms using different rewards as a function of stopping time K for different values of correlation coefficient ρ.

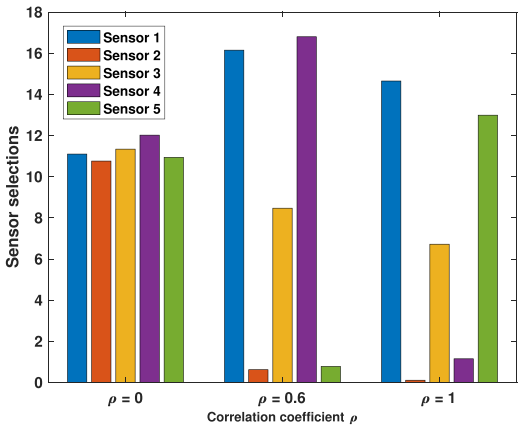
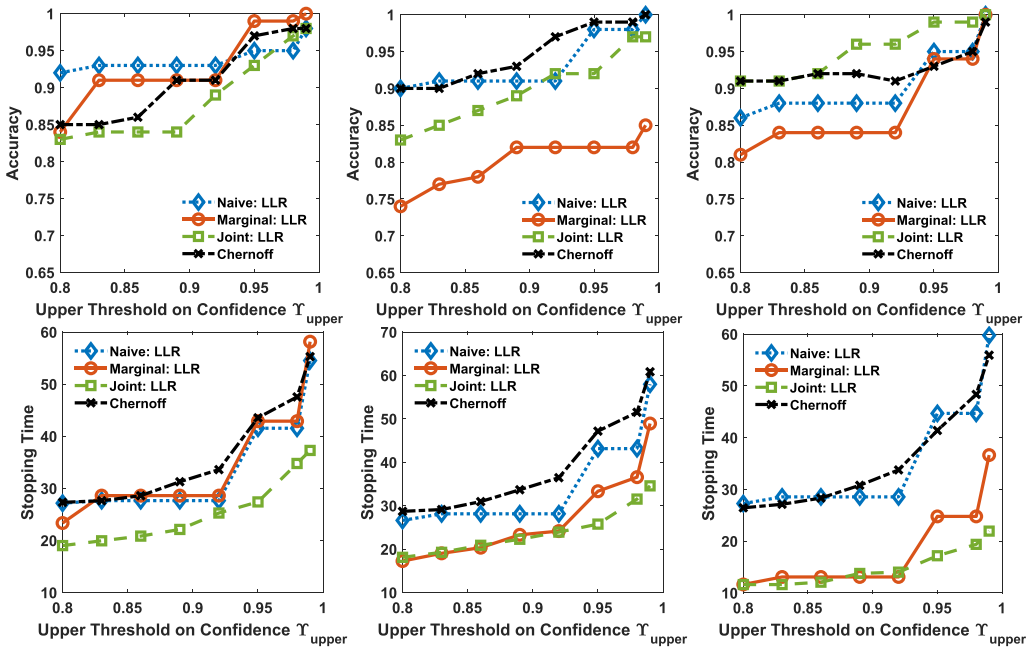


Fig. 2. Bar plot showing the number of times each sensor is chosen by our entropy-based anomaly detection algorithm for different values of correlation coefficient ρ when $\Upsilon_{\text{upper}} = 0.99$.

to [32] for similar figures for the entropy-based reward function showing similar trends under all settings. The accuracy and the stopping time of all the algorithms increase with Υ_{upper}. This trend is because as Υ_{upper} increases, the decision-maker requires more observations to satisfy the higher desired confidence level. In Fig. 4, we plot the accuracy and stopping time as a function of ρ. Comparing the two rewards, we infer that the performances of the LLR-based and entropy-based schemes are similar.

From Fig. 4 and along with the results in Fig. 1, and 3, we next look at the dependence of the algorithm performance on ρ. We notice that the accuracy of our algorithm is comparable to the other algorithms when $\rho = 0$ and $\rho = 1$. The accuracy degrades as ρ is close to 0.5. This behavior is because our algorithm uses approximate marginal probabilities to compute the confidence level whereas the other algorithms use exact values. This approximation in (23) is exact when $\rho = 0$ and $\rho = 1$. As ρ approaches 0.5, the approximation error increases, and the accuracy decreases. Also, the stopping times of the three algorithms are similar when $\rho = 0$. This behavior is because when $\rho = 0$, all the processes are independent, and the updates of our algorithm are exact. The naive marginal pmf-based algorithm and the Chernoff test also offer good performance when $\rho = 0$ as there is no underlying statistical dependence among the processes. Further, the stopping times of our algorithm and the joint pmf-based algorithm improve with ρ. As ρ increases, the processes become more correlated, and therefore, an observation corresponding to one process has more information about the other correlated processes. However, the naive marginal pmf-based algorithm ignores this correlation and handles the observations corresponding to the different processes independently. Therefore, its stopping time is insensitive to ρ and the performance is similar to that of the Chernoff test. Consequently, the difference between the stopping times of these two algorithms and our method (and joint pmf-based scheme) increase as ρ increases.

Furthermore, to gain insights into the selection rule of our algorithm, we visualize the actions decided by the algorithm with the entropy-based cost using a bar plot shown in Fig. 2.



(a) $\rho = 0$ (b) $\rho = 0.6$ (c) $\rho = 1$

Fig. 3. Performances of the centralized anomaly detection algorithms (using LLR-based reward for deep actor-critic methods) as a function of Υ_{upper} for different values of correlation coefficient ρ.

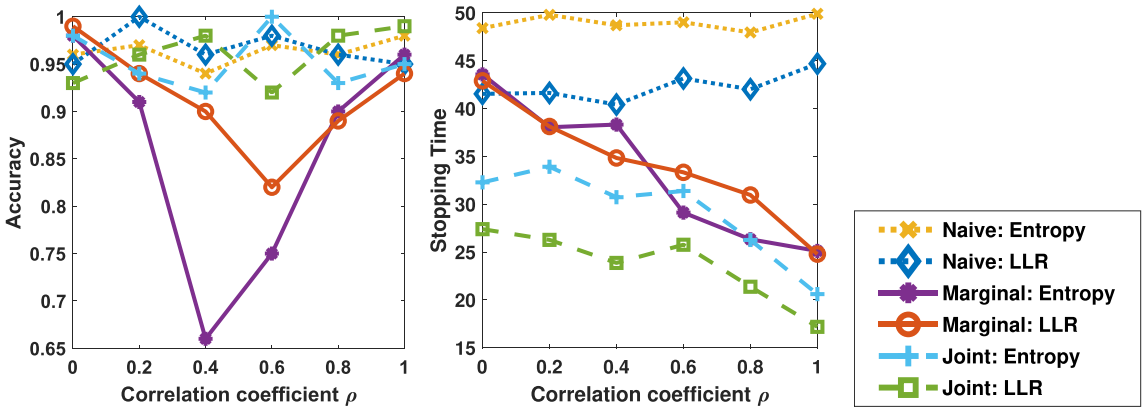


Fig. 4. Performances of different centralized deep actor-critic algorithms with varying ρ when $\Upsilon_{\text{upper}} = 0.95$.

When $\rho = 0$, all the processes are independent; therefore, all the sensors are chosen roughly the same number of times. When $\rho = 0.6$, processes $\{1, 2\}$ are correlated, so the algorithm observes the first process more often while reducing the number of times the second process's state $\boldsymbol{s}_2$ is observed. This behavior is because an observation from the first process also gives some information about the second process, and consequently, we observe the second process fewer times. The same behavior applies to the forth and fifth sensors, observing correlated processes. Also, when $\rho = 1$, we have $\boldsymbol{s}_1 = \boldsymbol{s}_2$. So, the first sensors gives more information about the second process $\boldsymbol{s}_2$ compared to the $\rho = 0.6$ case. Thus, the number of selections of the second sensor is further reduced. Since the third sensor observes $\boldsymbol{s}_3$, which is independent of others, its fraction is least affected by the variation in ρ. Hence, the algorithm adapts to the correlation coefficient, and as shown in Fig. 4, the stopping time and the total number of observations reduce with ρ. We make similar observations about the algorithm with LLR-based cost, whose plot is omitted to avoid repetition.

The algorithm run times given in Table I demonstrate that the joint pmf-based algorithm is computationally heavier (40% higher) compared to the other two algorithms. This observation is in agreement with our complexity analysis in Section IV-C. We also recall that the difference in the runtimes of the joint pmf-based algorithm and our algorithm grows with N.

Thus, our algorithm combines the best of two worlds by benefiting from the statistical dependence (like the joint scheme) and offering low-complexity (like the naive scheme).

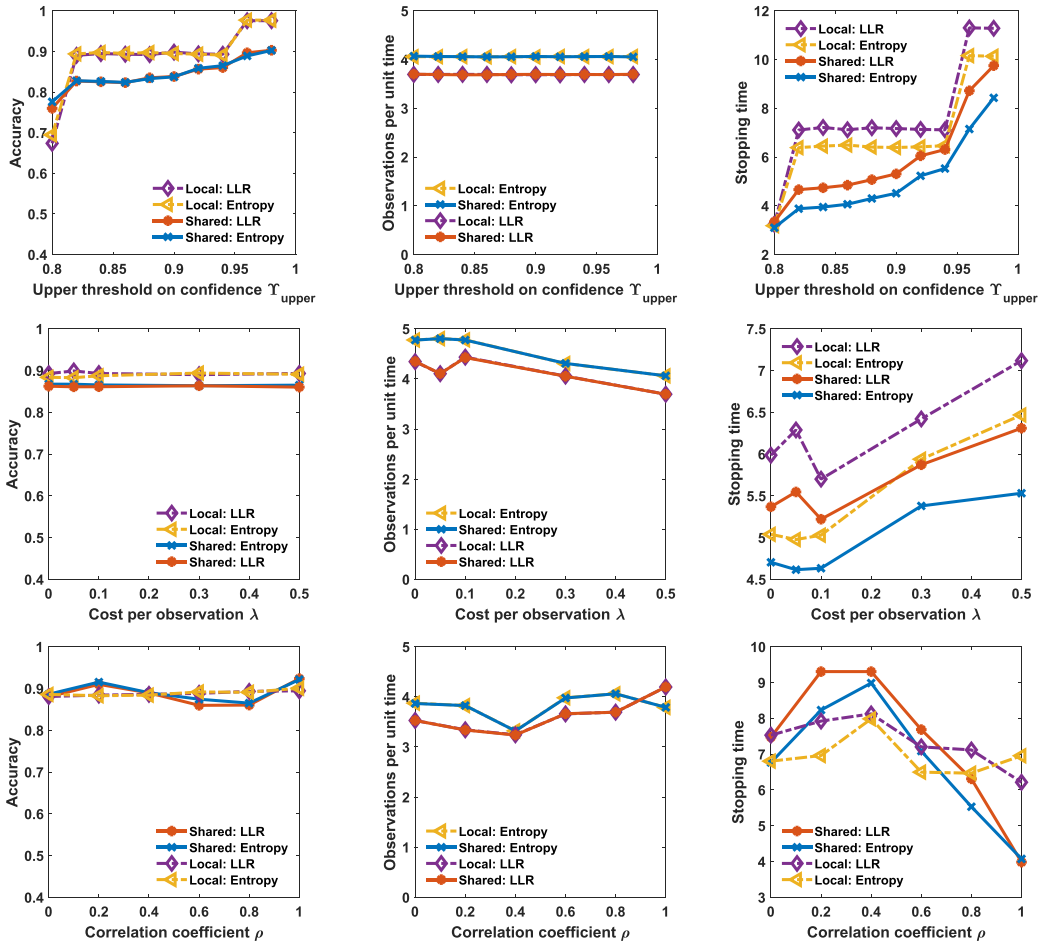


Fig. 5. Performances of the shared and local detection algorithms as a function of the upper threshold on confidence Υ_{upper}, sensing cost per observation λ, and correlation coefficient ρ with $N = 5$ processes. Unless mentioned otherwise, $\Upsilon_{\text{upper}} = 0.95$, $\rho = 0.8$ and $\lambda = 0.5$.

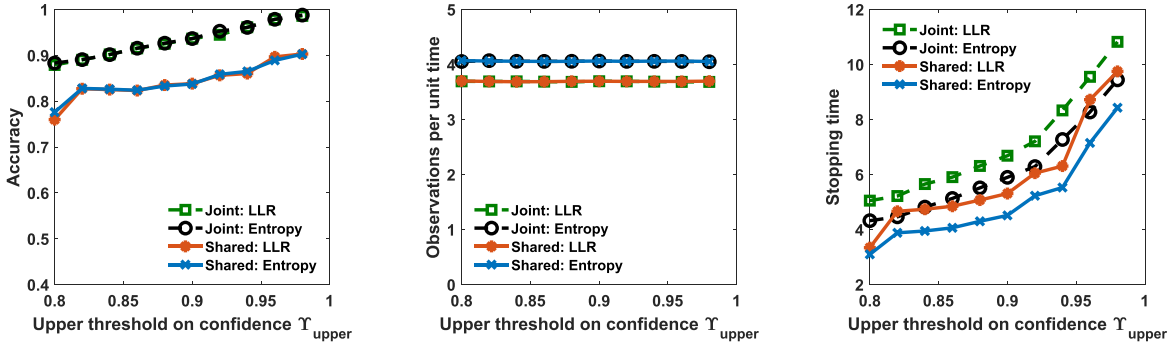


Fig. 6. Performances of the shared and joint detection algorithms as a function of the upper threshold on confidence Υ_{upper} when sensing cost per observation $\lambda = 0.5$ and correlation coefficient $\rho = 0.8$ with $N = 5$ processes.

B. Decentralized Algorithm

The architecture and parameters for our decentralized algorithm are as follows. We implement the actor and critic neural networks with four layers and the ReLU activation function between consecutive layers. The output layer of the actor layer uses the sigmoid function to ensure that the entries of the output vector of probabilities belong to [0, 1]. The parameters of the neural networks are updated using Adam Optimizer. We set the learning rates of the actor for the entropy and LLR-based reward functions as 2×10^{-5} and 3×10^{-5}, respectively. Also, the critic learning rate is 1×10^{-4} for both reward functions. Additionally, we set the discount factor as $\gamma = 0.9$ and the regularizers as $\eta = 1$ for the LLR-based reward in (11) and $\eta = 0.1$ for the entropy-based reward in (10).

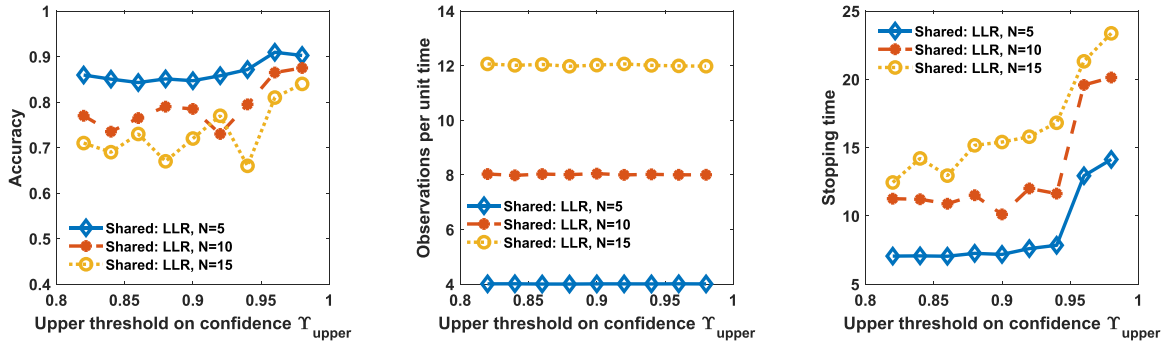


Fig. 7. Performances of the LLR-based shared detection algorithm with varying number of processes N as a function of the upper threshold on confidence Υ_{upper} when sensing cost per observation $\lambda = 0.5$ and correlation coefficient $\rho = 0.8$.

The numerical results for the decentralized algorithm are provided in Fig. 5–7. In the first setting, all the sensors share their observations with all the other sensors, i.e., $\mathcal{N}_i = \{1, 2, \ldots, N\}, \forall i$, which we refer to as the *shared detection algorithm*. In the second setting, none of the sensors share their observations with any other sensor, i.e., $\mathcal{N}_i = \{i\}, \forall i$, which we refer to as the *local detection algorithm*. We also consider a scheme with the joint pmf-based stopping rule (referred to as *joint detection algorithm*). The joint detection algorithm is identical to the shared detection algorithm, but its stopping rule is based on the joint posterior probabilities. This method computes the joint posterior probabilities $\pi(k) \in [0, 1]^{2^N}$ of all the entries of $\boldsymbol{s} \in [0, 1]^N$ (see [9] for details).

The first columns of Figs. 5 and 6 shows that the accuracy of all the algorithms increases with Υ_{upper}. However, the accuracy is insensitive to λ and ρ because the accuracy depends on the stopping rule that is independent of λ and ρ. Since the computations of the confidence used by the stopping rule are identical for the two versions (entropy and LLR-based schemes) of each algorithm (shared, local, and joint detection algorithms), they have the same accuracy levels. The accuracies of the local and joint detection algorithms are slightly higher than our shared detection algorithm, obtained at the cost of a higher stopping time.

The middle subfigure of the first row of Fig. 5 shows that, unlike the accuracy, the number of observations per unit time is insensitive to the Υ_{upper}. This trend is because the number of observations per unit time only depends on the selection policy learned by the algorithm, which in turn, depends only on the correlation coefficient ρ and sensing cost per observation λ. We see that the number of observations per unit time decreases with λ. This behavior naturally follows from (17) because the second term corresponding to the number of observations in (17) gets more (negative) weight in the reward function compared to the first term (entropy or LLR term). We also note that the number of observations per unit time does not show any noteworthy change as ρ varies. Further, due to the common centralized training, the selection policies of all the algorithms with a common reward function are the same. Hence, all the algorithms with a common reward function have the same number of observations per unit time.

The most sensitive performance metric is the stopping time which depends on both policy and stopping rule. The last columns of Figs. 5 and 6 indicate that the stopping times increase with Υ_{upper} and λ. For a given policy, larger Υ_{upper} implies a higher accuracy level which leads to a longer stopping time. Similarly, a larger λ results in a small number of observations per unit time, and consequently, stopping time increases with λ. This behavior captures the trade-off between the sensing cost and stopping time because the sensing cost decreases with λ. Further, the variation of the stopping time with ρ depends on the algorithm. The local detection algorithm ignores the correlation between the processes, and its stopping time does not vary significantly with ρ. However, the marginal probability updates of the shared and joint detection algorithms account for the learned statistical dependence among the processes. As ρ increases, each observation from a process gives more information on the other correlated processes, leading to a shorter stopping time.

Comparing the entropy-based and LLR-based algorithms, we see that both schemes offer almost the same level of accuracies and observations per unit time. However, the stopping time corresponding to the entropy-based scheme shows an improvement over the LLR-based schemes.

Finally, Fig. 7 shows the algorithm performance when the number of processes scales as $N = 5c$ where $c = 1, 2, 3$. Here, we have $2c$ pairs of statistically dependent processes captured by ρ and c independent processes. We observe similar performance trends with varying Υ_{upper} for all values of N. The observations per unit time and stopping time naturally increase with N. However, the accuracy slightly decreases with N for the same value of Υ_{upper}. This behavior is because Υ_{upper} defines the accuracy of correctly detecting a given process, and hence, the overall accuracy of detecting all the processes correctly decreases with N.

VII. Conclusion

We presented low-complexity centralized and decentralized algorithms to detect anomalous processes among a set of binary processes by dynamically selecting processes to be observed. The sequential process selection problem was formulated utilizing an MDP whose reward is defined using the entropy or

LLR of the marginal probabilities of the binary processes. The optimal process selection policy was obtained via the deep actor-critic RL algorithm that maximizes the long-term average reward of the MDP. The centralized algorithms chooses one process per unit time, whereas for the decentralized algorithm controls the number of observations per unit time via the sensing cost incorporated into the MDP reward. The numerical results showed that our algorithms offered good detection accuracy and stopping time while operating with low complexity. The algorithms also exploited the underlying statistical dependence among the binary processes, which led to a shorter stopping time when the processes were highly correlated. However, for scalable computing, our algorithms rely on approximate marginal probabilities. Quantifying this approximation error is an interesting direction for future work.

APPENDIX
DERIVATION OF POSTERIOR UPDATE

As observations arrive, we compute the belief vector as

$$\boldsymbol{\sigma}_i(k) = \mathbb{P}\left[\boldsymbol{s}_i = 0 \middle| \left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k}\right] \tag{19}$$

$$= \frac{\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = 0\right]\mathbb{P}\left[\boldsymbol{s}_i = 0\right]}{\sum_{s=0,1}\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = s\right]\mathbb{P}\left[\boldsymbol{s}_i = s\right]}, \tag{20}$$

for $i = 1, 2, \ldots, N$.

To further simplify the above relation, we take a close look at the term $\mathbb{P}[\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\}_{l=1}^{k}|\boldsymbol{s}_i = s]$. From (2), we have

$$\begin{aligned}
&\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = s\right] \\
&= \sum_{\boldsymbol{s}:\boldsymbol{s}_i=s} \mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}\right]\mathbb{P}\left[\boldsymbol{s}|\boldsymbol{s}_i = s\right] \\
&= \sum_{\boldsymbol{s}:\boldsymbol{s}_i=s} \mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1} \middle| \boldsymbol{s}\right]\prod_{a\in\mathcal{A}(k)}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right]\mathbb{P}\left[\boldsymbol{s}|\boldsymbol{s}_i = s\right] \\
&= \sum_{\boldsymbol{s}:\boldsymbol{s}_i=s}\prod_{a\in\mathcal{A}(k)}\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1}, \boldsymbol{s}\middle|\boldsymbol{s}_i = s\right]\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right] \\
&= \sum_{\boldsymbol{s}_{\mathcal{A}(k)}:\boldsymbol{s}_i=s}\prod_{a\in\mathcal{A}(k)}\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1}, \boldsymbol{s}_{\mathcal{A}(k)}\middle|\boldsymbol{s}_i = s\right] \\
&\quad\times\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right].
\end{aligned}$$

We recall the following algebraic identity,

$$\sum_{j=1}^{P} x_j \sum_{j=1}^{P} y_j - \sum_{j=1}^{P} x_j y_j = \sum_{j\neq k} x_j y_k \geq 0,$$

for any numbers $x_i, y_i \geq 0$ and holds with equality when the sequence satisfies $\sum_{j\neq k} x_j y_k = 0$. Applying this identity recursively, we deduce that

$$\begin{aligned}
&\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = s\right] \\
&\leq \sum_{\boldsymbol{s}_{\mathcal{A}(k)}:\boldsymbol{s}_i=s}\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1}, \boldsymbol{s}_{\mathcal{A}(k)}\middle|\boldsymbol{s}_i = s\right] \\
&\quad\times\prod_{a\in\mathcal{A}(k)}\sum_{\boldsymbol{s}_a:\boldsymbol{s}_i=s}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right].
\end{aligned}$$

To simplify our computations, we approximate the probability terms in (20) with the above upper bound to get

$$\begin{aligned}
\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = s\right] &\approx \mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1} \middle| \boldsymbol{s}_i = s\right] \\
&\quad\times\prod_{a\in\mathcal{A}(k)}\sum_{\boldsymbol{s}_a:\boldsymbol{s}_i=s}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right].
\end{aligned} \tag{21}$$

The above approximation holds with equality when $|\{\boldsymbol{s}_{\mathcal{A}(k)} : \boldsymbol{s}_i = s\}| = 1$, i.e., when $\boldsymbol{s}_{\mathcal{A}(k)}$ is completely determined by $\boldsymbol{s}_i$. Further, we have

$$\begin{aligned}
\sum_{\boldsymbol{s}_a:\boldsymbol{s}_i=s}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right] &\geq \sum_{\boldsymbol{s}_a:\boldsymbol{s}_i=s}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_a\right]\mathbb{P}\left[\boldsymbol{s}_a\middle|\boldsymbol{s}_i = s\right] \\
&= \mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_i = s\right].
\end{aligned} \tag{22}$$

The above bound holds with equality if $\mathbb{P}[\boldsymbol{s}_a|\boldsymbol{s}_i = s] \in \{0, 1\}$, i.e., $\boldsymbol{s}_a$ is completely determined from $\boldsymbol{s}_i$. We then obtain the following approximation from (21) and (22) for simplification,

$$\begin{aligned}
\mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k} \middle| \boldsymbol{s}_i = s\right] &\approx \mathbb{P}\left[\left\{\boldsymbol{y}_{\mathcal{A}(l)}(l)\right\}_{l=1}^{k-1} \middle| \boldsymbol{s}_i = s\right] \\
&\quad\times\prod_{a\in\mathcal{A}(k)}\mathbb{P}\left[\boldsymbol{y}_a(k)\middle|\boldsymbol{s}_i = s\right].
\end{aligned} \tag{23}$$

From (2), the observation $\boldsymbol{y}_{\mathcal{A}(k)}(k)$ is independent of all other observations, conditioned on the value of $\boldsymbol{s}_{\mathcal{A}(k)}$. As discussed above, the approximation is exact when $\boldsymbol{s}_{\mathcal{A}(k)}$ is a deterministic function of $\boldsymbol{s}_i$. Combining (20) and (23), we arrive at (5), completing the derivation.

REFERENCES

[1] W.-Y. Chung and S.-J. Oh, "Remote monitoring system with wireless sensors module for room environment," *Sensors Actuators B: Chem.*, vol. 113, no. 1, pp. 64–70, Jan. 2006.

[2] A. Bujnowski, J. Ruminski, A. Palinski, and J. Wtrorek, "Enhanced remote control providing medical functionalities," in *Proc. IEEE 7th Int. Conf. Pervasive Comput. Technol. Healthcare Workshops*, 2013, pp. 290–293.

[3] H. M. La, W. Sheng, and J. Chen, "Cooperative and active sensing in mobile sensor networks for scalar field mapping," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 45, no. 1, pp. 1–12, Jan. 2015.

[4] K. Zhu, A. Zhang, and D. Niyato, "Cost-effective active sparse urban sensing: An adversarial auto-encoder approach," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12064–12078, Aug. 2021.

[5] L. Lai, H. V. Poor, Y. Xin, and G. Georgiadis, "Quickest search over multiple sequences," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5375–5386, Aug. 2011.

[6] B. Hemo, T. Gafni, K. Cohen, and Q. Zhao, "Searching for anomalies over composite hypotheses," *IEEE Trans. Signal Process.*, vol. 68, pp. 1181–1196, 2020.

[7] M. Naghshvar and T. Javidi, "Active sequential hypothesis testing," *Ann. Statist.*, vol. 41, no. 6, pp. 2703–2738, Dec. 2013.
[8] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep actor-critic reinforcement learning for anomaly detection," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.
[9] G. Joseph, M. C. Gursoy, and P. K. Varshney, "Anomaly detection under controlled sensing using actor-critic reinforcement learning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun.*, 2020, pp. 1–5.
[10] H. Chernoff, "Sequential design of experiments," *Ann. Math. Statist.*, vol. 30, no. 3, pp. 755–770, Sep. 1959.
[11] S. Nitinawarat, G. K. Atia, and V. V. Veeravalli, "Controlled sensing for multihypothesis testing," *IEEE Trans. Autom. Control*, vol. 58, no. 10, pp. 2451–2464, Oct. 2013.
[12] S. A. Bessler, "Theory and applications of the sequential design of experiments, k-actions and infinitely many experiments. Part I. theory," Appl. Math. Statist. Labs, Stanford Univ., Stanford, CA, USA, Tech. Rep. SOL ONR 55, 1960.
[13] M. Naghshvar and T. Javidi, "Extrinsic Jensen-Shannon divergence with application in active hypothesis testing," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, 2012, pp. 2191–2195.
[14] M. Franceschetti, S. Marano, and V. Matta, "Chernoff test for strong-or-weak radar models," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 289–302, Jan. 2017.
[15] B. Huang, K. Cohen, and Q. Zhao, "Active anomaly detection in heterogeneous processes," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2284–2301, Apr. 2019.
[16] A. Gurevich, K. Cohen, and Q. Zhao, "Sequential anomaly detection under a nonlinear system cost," *IEEE Trans. Signal Process.*, vol. 67, no. 14, pp. 3689–3703, Jul. 2019.
[17] D. Chen, Q. Huang, H. Feng, Q. Zhao, and B. Hu, "Active anomaly detection with switching cost," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5346–5350.
[18] F. Qin, H. Feng, T. Yang, and B. Hu, "Low-cost active anomaly detection with switching latency," *Appl. Sci.*, vol. 11, no. 7, Jan. 2021, Art. no. 2976.
[19] F. Qin, D. Chen, H. Feng, Q. Zhao, T. Yang, and B. Hu, "Active anomaly detection with switching cost," 2021, *arXiv:1810.11800*.
[20] R. S. Blum, S. A. Kassam, and H. V. Poor, "Distributed detection with multiple sensors II. advanced topics," *Proc. IEEE*, vol. 85, no. 1, pp. 64–79, Jan. 1997.
[21] R. Viswanathan and P. K. Varshney, "Distributed detection with multiple sensors part I. fundamentals," *Proc. IEEE*, vol. 85, no. 1, pp. 54–63, Jan. 1997.
[22] Y. Mei, "Asymptotic optimality theory for decentralized sequential hypothesis testing in sensor networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2072–2089, May 2008.
[23] Y. Wang and Y. Mei, "Asymptotic optimality theory for decentralized sequential multihypothesis testing problems," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 7068–7083, Oct. 2011.
[24] Z. Chair and P. K. Varshney, "Distributed Bayesian hypothesis testing with distributed data fusion," *IEEE Trans. Syst. Man Cybern.*, vol. 18, no. 5, pp. 695–699, Sep./Oct. 1988.
[25] D. Pados, K. W. Halford, D. Kazakos, and P. Papantoni-Kazakos, "Distributed binary hypothesis testing with feedback," *IEEE Trans. Syst. Man Cybern.*, vol. 25, no. 1, pp. 21–42, Jan. 1995.
[26] A. Rangi, M. Franceschetti, and S. Marano, "Decentralized Chernoff test in sensor networks," in *Proc. IEEE Int. Symp. Inf. Theory*, 2018, pp. 501–505.
[27] D. Kartik, A. Nayyar, and U. Mitra, "Sequential experiment design for hypothesis verification," in *Proc. IEEE 52nd Asilomar Conf. Signals, Syst., Comput.*, 2018, pp. 631–635.
[28] D. Bertsekas, *Dynamic Programming and Optimal Control: Volume I*. Nashua, NH, USA: Athena Sci., 2012.
[29] D. Kartik, E. Sabir, U. Mitra, and P. Natarajan, "Policy design for active sequential hypothesis testing using deep learning," in *Proc. IEEE 56th Annu. Allerton Conf. Commun., Control, Comput.*, 2018, pp. 741–748.
[30] G. Joseph, C. Zhong, M. C. Gursoy, S. Velipasalar, and P. K. Varshney, "Anomaly detection under controlled sensing using actor-critic reinforcement learning," in *Proc. IEEE Globecom*, 2020, pp. 1–5.
[31] G. Joseph, M. C. Gursoy, and P. K. Varshney, "Temporal detection of anomalies via actor-critic based controlled sensing," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.
[32] G. Joseph, M. C. Gursoy, and P. K. Varshney, "A scalable algorithm for anomaly detection via learning-based controlled sensing," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

Geethu Joseph (Member, IEEE) received the B. Tech. degree in electronics and communication engineering from the National Institute of Technology, Calicut, India, in 2011, and the M. E. degree in signal processing and the Ph.D. degree in electrical communication engineering, from the Indian Institute of Science Bangalore, Bangalore, India, in 2014 and 2019, respectively. From 2019 to 2021, she was a postdoctoral Fellow with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA. She is currently an Assistant Professor with the signal processing systems group with the Delft University of Technology, Delft, The Netherlands. Her research interests include statistical signal processing, network control, and machine learning. Dr. Joseph was the recipient of the 2022 IEEE SPS best Ph.D. dissertation award and the 2020 SPCOM best doctoral dissertation award. She is an Associate Editor for the IEEE SENSORS JOURNAL.

Chen Zhong received the B.S. degree in information engineering from the Beijing Institute of Technology, Beijing, China, in 2014, and the M.S. degree in electrical engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2016, and the Ph.D. degree in electrical engineering from the Syracuse University, Syracuse, NY, USA, in 2022. Her research interests include the areas of wireless communication and networking, signal processing, and machine learning.

M. Cenk Gursoy (Senior Member, IEEE) received the B.S. degree (Hons.) in electrical and electronics engineering from Bogazici University, Istanbul, Turkey, in 1999, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2004. He is currently a Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA.

Senem Velipasalar (Senior Member, IEEE) received the B.S. degree in electrical and electronic engineering with high honors from Bogazici University, Istanbul, Turkey, in 1999, the M.S. degree in electrical sciences and computer engineering from Brown University, Providence, RI, USA, in 2001, the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 2007 and 2004, respectively. From 2007 to 2011, she was an Assistant Professor with the Department of Electrical Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA. She is currently a Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA.

Pramod K. Varshney (Life Fellow, IEEE) received the B.S. degree in electrical engineering and computer science (with Highest Hons.), in 1972, the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1974 and 1976, respectively. Since 1976, he has been with Syracuse University, Syracuse, NY, USA, where he is currently a Distinguished Professor of electrical engineering and computer science and the Director of CASE: Center for Advanced Systems and Engineering.