Monitoring State Transitions in Markovian Systems with Sampling Cost

Kumar Saurav
Shiv Nadar Institution of Eminence
Delhi-NCR, India
kumar.saurav@snu.edu.in

Ness B. Shroff
The Ohio State University
Columbus, Ohio, USA
shroff.11@osu.edu.in

Yingbin Liang
The Ohio State University
Columbus, Ohio, USA
liang.889@osu.edu.in

***Abstract*—We consider a node-monitor pair, where the node's state varies with time. The monitor needs to track the node's state at all times, however, there is a fixed cost for each state query. So the monitor may instead predict the state using time-series forecasting methods, including time-series foundation models (TSFMs), and query only when prediction uncertainty is high. Since query decisions influence prediction accuracy, determining when to query is nontrivial. A natural approach is a greedy policy that predicts when the expected prediction loss is below the query cost and queries otherwise. We analyze this policy in a Markovian setting, where the optimal (OPT) strategy is a state-dependent threshold policy minimizing the time-averaged sum of query cost and prediction losses. We show that, in general, the greedy policy is suboptimal and can have an unbounded competitive ratio, but under common conditions—such as identically distributed transition probabilities—it performs close to OPT. For the case of unknown transition probabilities, we further propose a projected stochastic gradient descent (PSGD)-based learning variant of the greedy policy, which achieves a favorable predict–query tradeoff with improved computational efficiency compared to OPT.**

***Index Terms*—Markov chain, scheduling, remote estimation, age of information.**

I. Introduction

Consider a status update system, where a remote monitor needs to track the time-varying state of a node. Traditionally, in such systems, the monitor repeatedly queries the node, and maintains an accurate estimate of its current state at all times. However, with advent of large scale energy-constrainted IoT systems, it is critical to optimize the energy and channel usage. Therefore, in the modern paradigm, the monitor must use the available information to predict node's current state, and query only when the uncertainty in node's state is large. In this paper, we refer to it as *interleaved query-predict approach (IQP)*.

IQP raises two coupled questions: (i) how to predict, and (ii) when to query. A natural greedy rule predicts the state with minimum expected loss and queries only when that loss exceeds the query cost. However, this ignores the informational value of queries: a query can reduce future prediction errors, so querying now may lower long-run cost even if its immediate cost exceeds the current expected loss. While the greedy policy is simple and attractive under model uncertainty, an optimal policy may be computationally costly. Hence we seek policies that balance near-optimal query costs and prediction loss with practical implementability. Therefore, the decision problem needs to be analyzed thoroughly to find a balanced solution that is near-optimal in query cost and prediction loss, and easily implementable for wider adoption.

With this motivation, in this paper, we consider the greedy policy introduced above, and compare its performance in relation to an optimal policy (OPT) that incurs minimum average cost (weighted sum of average query cost and average prediction loss). We seek to identify scenarios where the greedy policy (which is simple to implement) is a promising alternative to OPT in average cost. To this end, we consider a setting where the node's state varies following a finite-state Markov chain, and analyze the policies for scenarios where the state transition probabilities are i) known, and ii) unknown.

The reason we consider a Markovian system is two-folds. First, it provides a structured setting where we can derive OPT and analyze its properties in relation to the greedy policy. Second, in this setting, OPT can make long term optimizations, and therefore has most advantage over the greedy policy. Thus, intuitively, it provides a bound on the optimality gap for the greedy policy in generic settings.

Further, we adopt a linear combination of prediction loss and query cost as the objective, as it compactly captures the query–predict tradeoff and extends naturally to multi-node settings. (i) With suitable weights [1], this formulation balances inversely related costs while satisfying all constraints. (ii) It further allows derivation of Whittle's index (WI) [2], enabling Whittle's index policy (WIP) for efficient multi-node query scheduling to minimize average prediction error.

Related works. Prior works on remote tracking have mostly focused on minimizing state error using metrics like age-of-information (AoI) [3], [4], age-of-incorrect-information (AoII) [5], and others [6]. These approaches typically optimize query timing to reduce delay between state changes and subsequent queries, subject to query cost constraints. However, treating all state errors equally can be misleading. For instance, in a safety application, failing to detect a fire is far more costly than a false alarm. Ignoring such inter-state criticality may lead to severe consequences.

Another variant of remote tracking problem considered in literature is on estimation of remote process, where the goal is to minimize the mean squared error between the actual and the estimated process [7], [8]. These works assume that the process follows simple dynamics such as Wiener or Ornstein-Uhlenbeck process. Therefore, the estimation error is

correlated to the time since the last query, and minimizing the mean squared error reduces to minimizing AoI. Note that such an approach is not suitable for discrete state spaces where state transitions may be sudden, and cost due to state errors may vary significantly across different states.

Most relevant work on remote tracking with inter-state criticality is [9], which considers a multi-node setting where the state of the node varies following a finite-state Markov chain, and in each time slot, the monitor must choose the subset of nodes to query such that, under a constraint on the maximum number of simultaneous queries, the average state error across all nodes is minimized. With some relaxations, this problem reduces to the single-node setting considered in this paper, and assuming the state transition probabilities are known, [9] models it as a Markov decision process (MDP), and solves it using value iteration. However, this needs to be repeated for each time slot, making the solution computationally expensive. Hence, we need a simpler alternative such as a greedy policy.

An important aspect of the considered problem is the knowledge of the state transition probability matrix $\mathcal{P}$ which determines the uncertainty in future states. Though prior works like [7]–[9] assume $\mathcal{P}$ to be known, in practice, $\mathcal{P}$ may have to be inferred/learnt from observations. A popular approach is to use the classical stopping-time technique [10] where a policy queries in first α time slots (for some $\alpha > 0$) and estimate one-step transition probability $\hat{p}_{ij}$ (from state s_i to s_j) by computing the ratio of number of transitions from state s_i to s_j and the number of visits to state s_i. However, as we discuss in Section IV, this approach is inefficient for estimating $n-$step transition probability ($n \gg 1$) as is required for the considered problem. This adds another dimension to the considered problem calling for a more efficient approach to learn $\mathcal{P}$ with only few intermittent queries.

Contributions. In this paper, we derive OPT for the considered problem with known $\mathcal{P}$, and compare the greedy policy π^g (and some other heuristic policies) with OPT. We show that in certain scenarios π^g's performance can be arbitrarily bad compared to OPT, and propose a fix for such an issue. We also identify settings where π^g has near-optimal performance, and is preferable over OPT due to ease of implementation (e.g. when transition probabilities are unknown).

II. System Model

Consider a discrete-time setting where the state of a node varies with time following some finite (discrete)-state Markov chain $\mathcal{M}$, and a monitor needs to track the node's current state at all times. To track the state, the monitor can either query (sample) the current state of the node, or predict it based on the previously queried state.

Let $\mathcal{S} = \{s_1, s_2, ..., s_K\}$ (for finite integer $K > 0$) denote the set of all possible states, and $\mathcal{P}$ be the state transition probability matrix for $\mathcal{M}$. At time t, if the current state of the node is $X_t \in \mathcal{S}$, and the monitor predicts the state to be $\hat{X}_t \in \mathcal{S}$, then there is a state error (prediction loss) equal to $L(X_t, \hat{X}_t) \geq 0$, where $L(\cdot,\cdot)$ is the loss function. If the monitor queries the state at time t, then the state error is 0, and the monitor receives the actual state X_t (i.e. $\hat{X}_t = X_t$). However, it incurs a fixed query (sampling) cost $c \geq 0$.

Remark 1: We assume that the loss $L(X_t, \hat{X}_t)$ is deterministic for all possible combinations of $X_t, \hat{X}_t \in \mathcal{S}$, and recorded as a loss matrix $\mathcal{L} \in \mathbb{R}_{\geq 0}^{K \times K}$, with $(i,j)-$th element $\ell_{ij} = L(s_i, s_j)\ \forall i, j \in \{1, 2, ..., K\}$.

The goal is to find an online (causal) policy π that at each time t, decides whether to query or predict, and if predict, then how to choose $\hat{X}_t$ such that the time-averaged sum of the expected prediction loss and the query cost (in short, *sum average cost* $\Gamma(\pi)$) is minimized. Succinctly, this is given as

$$\min_{\pi \in \Pi} \Gamma(\pi) \equiv \min_{\pi \in \Pi} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[L(X_t, \hat{X}_t^{\pi}) + c \cdot Q_t^{\pi}], \quad (1)$$

where Π denotes the set of all online policies π, while Q_t^{π} denotes the number of queries made until time t under policy π. In this paper, we consider two variants of the above problem: (i) when $\mathcal{P}$ is known at the monitor, and (ii) when $\mathcal{P}$ is unknown at the monitor. First, we focus on the former case, in Section III, and analyze the performance of a greedy policy π^g against an optimal policy $\pi^\star$ for (1). Subsequently, in Section IV we extend π^g to learn $\mathcal{P}$, and discuss its practicality over $\pi^\star$ due to ease of implementation and low-complexity.

Remark 2: In this paper, let $p_{ij}^{(n)}$ denote the n-step transition probability from state s_i to s_j, where $p_{ij}^{(1)} = p_{ij}$; equivalently, $p_{ij}^{(n)}$ is the (i,j)-th entry of $\mathcal{P}^n$, the n-th power of the transition matrix $\mathcal{P}$. Also, the hat symbol (e.g., $\hat{X}$) denotes an estimate or prediction of a variable, and $\tau(t)$ represents the most recent query time before t.

III. Case 1: Transition Probabilities are Known

At any time t, given $\mathcal{P}$ and the previously queried state $X_{\tau(t)}$ (at time $\tau(t)$) a critical question is: how to predict the current state X_t? We answer this in the following Lemma 1.

Lemma 1: Let $n_t = t - \tau(t)$, i.e. the time since last query until the current time t. Also, let the queried state at time $\tau(t)$ be s_i. At time t, the optimal prediction strategy is to pick state

$$s_{k^\star} = \arg\min_{s_k \in \mathcal{S}} \sum_{s_j \in \mathcal{S}} p_{ij}^{(n_t)} \cdot \ell_{jk}, \quad (2)$$

where ℓ_{jk} is the loss when the state is s_j while the prediction is s_k. Also, the optimal (expected) prediction loss is $\sum_{s_j \in \mathcal{S}} p_{ij}^{(n_t)} \ell_{jk^\star}$.

Proof: Let an optimal prediction policy π^p predicts state $\hat{X}_t = s_k$ with probability $q_{ik}^{(n_t)}$ ($\forall k \in \mathcal{S}$). Since the probability $P(X_t = s_j | X_{\tau(t)} = s_i) = p_{ij}^{(n_t)}$, the expected prediction loss for π^p is $\sum_{s_k \in \mathcal{S}} q_{ik}^{(n_t)} \cdot \sum_{s_j \in \mathcal{S}} p_{ij}^{(n_t)} \cdot \ell_{jk}$. Clearly, this expected prediction loss for π^p is minimum if $q_{ik}^{(n_t)} = 1$ for $s_k = s_{k^\star}$ (2), and $q_{ik}^{(n_t)} = 0$ for $s_k \neq s_{k^\star}$. Also, the corresponding minimum prediction loss is equal to $\sum_{s_j \in \mathcal{S}} p_{ij}^{(n_t)} \ell_{jk^\star}$. ∎

The next step is to design a policy that decides whether to predict (2) or query in each time slot such that the average sum cost (1) is minimized. To this end, a natural approach is to follow a greedy policy π^g (Algorithm 1) that at any time,

predicts if the expected prediction loss for (2) is less than the query cost c, and queries otherwise.

Algorithm 1 Greedy policy π^g.

let s_i denote the queried state at time $\tau(t)$, $\forall\ t \geq 1$;
for time $t = 1, 2, 3, \cdots$ **do**
  compute $n_t = t - \tau(t)$;
  if $\sum_{s_j \in \mathcal{S}} p_{ij}^{(n_t)} \ell_{jk^\star} < c$ **then**
    predict state $\hat{X}_t = s_k^\star$ (2);
  else
    query the current state X_t;
  end if
end for

Intuitively, it is always better to query if the expected prediction loss is higher than (or equal to) the query cost. However, under π^g, always predicting when the expected prediction loss is less than the query cost can be suboptimal. This is because by revealing current state, a query may help minimize future prediction losses as well, as shown next.

Theorem 1: When Markov chain $\mathcal{M}$ has absorbing states, or in general, states with very little uncertainty in future state trajectory (offering low prediction loss over a period of time), the ratio of the sum average cost (1) for π^g to that of an optimal policy $\pi^\star$ can be arbitrarily large.

Proof: Consider a node with states $\mathcal{S} = \{s_1, s_2, s_3\}$, and matrices $\mathcal{P} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $\mathcal{L} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$. Also, let the initial state $X_0 = 1$, and the query cost $c = 1$. In this setting, note that $\mathcal{P}$ is idempotent, i.e. $\mathcal{P}^n = \mathcal{P}$, and $\sum_{s_j \in \mathcal{S}} p_{1j}^{(n)} \ell_{jk^\star} = 0.5 < 1 = c,\ \forall\ n \geq 1$. Therefore, the greedy policy π^g never queries, and incurs an expected prediction loss of 0.5 in each time slot, leading to the sum average cost (1) equal to 0.5. However, note that $\mathcal{P}$ is such that starting in state $X_0 = 1$, in slot $t = 1$, the node transitions either to state 2 or state 3 with equal probability, and then remains in that state forever thereafter. Therefore, an optimal policy $\pi^\star$ queries in slot 1 (incurring a query cost $c = 1$), and then predicts in all future time slots as the prediction loss will be 0. Hence, as $t \to \infty$, the expected sum average cost (1) for $\pi^\star$ approaches 0. Thus, computing the ratio of the expected sum average costs (1) for π^g and $\pi^\star$, we get the result. ■

The question is, how π^g can be made robust against scenarios in Theorem 1, and in what scenarios is π^g close to optimal? We analyze this numerically in Section V by comparing the performance of π^g with $\pi^\star$, derived next.

An optimal online policy $\pi^\star$ for Markovian systems

Lemma 2: For the considered Markovian system, there exists an optimal online policy $\pi^\star$ such that i) $\pi^\star$ is stationary, and ii) if the latest query has been in slot τ, revealing the current state $X_\tau = s_i$, $\pi^\star$ queries next in slot $\tau + \mu_i^\star$, where $\mu_i^\star$ is a deterministic positive integer that only depends on s_i. Also, in each slot $t \in \{\tau + 1, \tau + 2, \cdots, \tau + (\mu_i^\star - 1)\})$, $\pi^\star$ predicts following the optimal prediction strategy (2).

Proof: In any slot τ, if the current state $X_\tau = s_i$ is known, then because of the Markovian property of state transition process, the distribution over the future state trajectories depends only on s_i (i.e. independent of the past states or the time index τ). Moreover, in subsequent time slots, until the state is queried again, there is no external input or additional information that can influence this distribution. Therefore, there must exist an optimal online policy $\pi^\star$ that chooses the next query time deterministically, depending only on s_i. This also implies that $\pi^\star$'s decision is independent of τ, i.e. $\pi^\star$ is stationary.

Further, since $\pi^\star$ does not query at time $t \in \{\tau + 1, \tau + 2, \cdots, \tau + (\mu_i^\star - 1)\}$ (by definition of $\mu_i^\star$), it must predict following the optimal prediction strategy (2) to minimize the expected prediction loss. ■

Lemma 2 implies that a state-dependent threshold policy $\pi \in \Pi$ (Algorithm 2) that at any time t, if the queried state is $s_i \in \mathcal{S}$, then predicts $k^\star$ (2) at times $t+1, t+2, \cdots, t+\mu_i^\pi-1$, and queries next at time $t + \mu_i^\pi$, is optimal for some threshold $\mu_i^\pi = \mu_i^\star$, $\forall i \in \mathcal{S}$. Therefore, we next derive $\mu_i^\star$'s.

Algorithm 2 State-dependent threshold policy $\pi \in \Pi$.

compute $\mu^\pi = \{\mu_i^\pi > 0 : \forall\ i \in \mathcal{S}\}$; // see Lemma 3
query initial state X_0;
$\hat{t} = \mu_{X_0}^\pi$; // compute next query time
for time $t = 1, 2, 3, \cdots$ **do**
  if $t \neq \hat{t}$ **then**
    predict state $\hat{X}_t = s_{k^\star}$ (2);
  else
    query current state X_t;
    $\hat{t} = t + \mu_{X_t}^\pi$; // update next query time
  end if
end for

For policy $\pi \in \Pi$, let τ_n^π denote the n^{th} query time, and $X_{\tau_n}^\pi$ be the corresponding queried state. Note that the time-axis can be partitioned into intervals between successive query times under π. We refer to each of these intervals as a *stage* for π, with the n^{th} stage $\chi_n^\pi = \{\tau_n^\pi, \cdots, \tau_{n+1}^\pi - 1\}$, where $\tau_{n+1}^\pi = \tau_n^\pi + \mu_{X_{\tau_n}^\pi}^\pi$. In each stage, there is exactly one query (at the start of the stage), followed by prediction in the remaining time slots of the stage. Also, due to the Markovian property, under policy π, the prediction losses in the stage only depends on the state s_i revealed after the query (*queried state*). Therefore, we denote the expected total cost incurred in a stage with queried state s_i as $\mathbb{E}[\Lambda(s_i, \mu_i^\pi)]$, where μ_i^π is the threshold for state s_i under state-dependent threshold policy π. Furthermore, the relative cost in the stage for policy π (relative to optimal policy $\pi^\star$) is $\mathbb{E}[\overline{\Lambda}^\pi(s_i)] = \mathbb{E}[\Lambda(s_i, \mu_i^\pi)] - \Gamma(\pi^\star)\mu_i^\pi$, where $\Gamma(\pi^\star)$ is the sum average cost (1) for an optimal policy $\pi^\star$.

Note that under policy π, the queried state $X_{\tau_n^\pi}$ in i^{th} stage only depends on the queried state s_i in the previous stage. In particular, $X_{\tau_n^\pi} = s_j$ with probability $p_{ij}^{(\mu_i^\pi)}$, where the superscript (μ_i^π) denotes μ_i^π step transition probability. Therefore, the relative cost $h^\pi(s_i)$ incurred under policy π starting from the stage with queried state s_i is

$$h^{\pi}(s_i) = \mathbb{E}[\Lambda(s_i, \mu_i^{\pi})] - \Gamma(\pi^{\star})\mu_i^{\pi} + \sum_{j \in \mathcal{S}} p_{ij}^{(\mu_i^{\pi})} h^{\pi}(s_j). \quad (3)$$

Without loss of generality, let the queried state in first/initial stage be s_1. Then under an optimal policy $\pi^{\star}$, $h^{\pi^{\star}}(s_1) = 0$. Using this, we get the following result.

Lemma 3: A policy $\pi \in \Pi$ is an optimal solution of problem (1) if and only if $h^{\pi}(s_1) = 0$.

Note that in $h^{\pi}(s_1)$ there are $K+1$ unknowns: μ_i^{π}'s (for $i = 1, 2, \cdots, K$) and $\Gamma(\pi^{\star})$ (other terms including $h^{\pi}(s_i)$'s are functions of these unknowns). Thus, combining the optimality condition $h^{\pi}(s_1) = 0$ along with the K equations in (3) (for $i = 1, 2, \cdots, K$), and using iterative methods, such as policy iteration [11], we get $\mu_i^{\star}$'s.

Theorem 2: On solving $h^{\pi}(s_1) = 0$ along with the K equations in (3) (for $i = 1, 2, \cdots, K$) using policy iteration [11], we get the optimal thresholds $\mu_i^{\star}$'s for the state-dependent threshold policy (Algorithm 2), thus, revealing $\pi^{\star}$.

IV. Case 2: Transition Probabilities are Unknown

In previous section, we assumed the state transition probability matrix $\mathcal{P}$ to be known, and used it to compute the optimal prediction loss in Lemma 1, and formulate the relative cost function (3). However, in practice, $\mathcal{P}$ may be unknown. In such cases, one possible option is model-free learning where a policy chooses actions to minimize the sum average cost (1) directly by trial-and-error without explicitly learning $\mathcal{P}$. However, this approach suffers a critical challenge because the loss is not revealed in case of a prediction. Moreover, the cost function (prediction loss (2)) at time t depends on $t - \tau(t)$ (the time since last query). Therefore, under the model-free learning approach, a policy needs to learn to optimize the cost function (2) for each time slot $\tau + 1, \tau + 2, \cdots$, separately.

Alternatively, a policy could learn the probability matrix $\mathcal{P}$ (estimate $\hat{\mathcal{P}}$) and use it to compute the cost function (2) (using $(\hat{\mathcal{P}})^n$) for all time slots $\tau+n$ ($n \geq 1$) as in the previous section. However, to compute $\hat{\mathcal{P}}$, the classical stopping-time technique [10] where a policy queries in first α time slots (for some $\alpha > 0$) and computes probability $\hat{p}_{ij}$ by computing the ratio of number of transitions from state s_i to s_j and the number of visits to state s_i is inefficient. This is because the error in $\hat{\mathcal{P}}$ amplifies while computing $\hat{\mathcal{P}}^2, \hat{\mathcal{P}}^3, \cdots$ (at time $\tau+2, \tau+3, \cdots$ respectively), thus requiring α to be very large. Therefore, in this section, we propose an alternate approach for updating $\hat{\mathcal{P}}$ using *projected stochastic gradient descent (PSGD)* [12].

Broadly, the PSGD-based approach (Algorithm 3) is as follows: Whenever there is a query at time $\tau+n$ ($\forall\, n \geq 1$), we compute a loss as a function of the state predicted using current estimate $(\hat{\mathcal{P}})^n$ and the queried state $X_{\tau+n}$. Subsequently, we compute the gradient of the loss with respect to (w.r.t.) $\hat{\mathcal{P}}$, and update the current estimate of $\hat{\mathcal{P}}$ using the gradient. Since the row-sum of matrix $\hat{\mathcal{P}}$ must be 1, we finally project $\hat{\mathcal{P}}$ (reset its negative elements to zero, and then normalize each element by the row-sum) to ensure that the row-sum constraint is satisfied.

Algorithm 3 PSGD to update $\mathcal{P}$ estimate after each new query.

latest query time τ and queried state $X_\tau = s_i$;
for time $t = \tau + 1, \tau + 2, \tau + 3, \cdots$ **do**
  if query at time t with state $X_t = s_j$ **then**
    define $y_k = 1$ if $k = j$ and $y_k = 0$ otherwise;
    compute $\hat{p}_{ik}^{(n)}$, $\forall k$, using current estimate $\hat{\mathcal{P}}$;
    compute loss $\mathcal{F}_{i,n} = \sum_{k=1}^{K} \left(y_k - \hat{p}_{ik}^{(n)}\right)^2$;
    compute gradient $\nabla \mathcal{F}_{i,n}$ w.r.t. $\hat{\mathcal{P}}$;
    update $\hat{\mathcal{P}} = \hat{\mathcal{P}} - \eta_t \nabla \mathcal{F}_{i,n}$, where η_t is the learning rate;
    project $\hat{\mathcal{P}}$ so that the row-sum of $\hat{\mathcal{P}}$ is 1;
    update $\tau = t$ and $X_\tau = s_j$;
  else
    continue;
  end if
end for

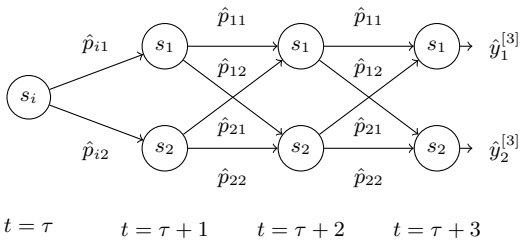

Figure 1: A neural network representation of the state transition process. This graph can be used to compute the error in state predictions, as well as the gradient of the loss w.r.t. probability estimates $\hat{p}_{ij}$.

Remark 3: The gradients of the loss $\mathcal{F}_{i,n}$ w.r.t. $\hat{p}_{ij}$ can be computed efficiently using the chain rule. In interpreting the state transitions over time as a feedforward neural network (Figure 1), where each node in the m^{th} layer ($m = 0, 1, \cdots, n$) represents the state of the Markov chain at time $t = \tau + m$, and the edges represent the transition probabilities, we can compute the loss $\mathcal{F}_{i,n}$ as a function of the estimated transition probabilities $\hat{p}_{ij}$. The gradient of the loss w.r.t. $\hat{p}_{ij}$ ($\forall\, m = 1, 2, \cdots, n$) can be computed using backpropagation. Note that the gradient w.r.t. $\hat{p}_{ij}$ in each layer can be different. The actual gradient of the loss $\mathcal{F}_{i,n}$ w.r.t. $\hat{p}_{ij}$ is equal to the sum of the gradients of the loss w.r.t. $\hat{p}_{ij}$ in each layer.

A major advantage of the PSGD approach is that it allows querying in non-consecutive time slots. Note that if the state at time τ is s_i and we query at time $\tau + 1$, the new state only reveals information about the transition probabilities p_{ij} from state i. Therefore, despite being extremely useful in computing initial estimates of $\mathcal{P}$, they turn out expensive (require too many queries) while refining transition probability estimates, especially from states that are visited less frequently. In contrast, note that $p_{1j}^{(n)}$ for $n \geq 3$ depends on all elements of $\mathcal{P}$. Hence, using PSGD we can query at any time $\tau + n$ for $n \geq 3$, and refine p_{ij} estimates $\forall\, i, j = \{1, 2, \cdots, K\}$, easily.

Remark 4: We use PSGD to minimize loss in transition

probability estimates $\hat{\mathcal{P}}$ instead of prediction loss (2) because the prediction loss is a function of $t-\tau$ (time since last query), and the minimizer of prediction loss can vary with time. In contrast, the minimizer of loss $\mathcal{F}_{i,n}$ is $\mathcal{P}$ for all $n \geq 1$.

Though PSGD approach is useful in estimating the transition probability matrix $\mathcal{P}$, the overall performance in minimizing the sum average cost (1) still depends on when the queries are made. Therefore, we must combine PSGD with some policy that at each time, uses the $\hat{\mathcal{P}}$ estimates computed using PSGD and make query/predict decision optimizing (1). However, this could be a risky approach because if a crude probability estimate $\hat{\mathcal{P}}$ suggests to never query, then $\hat{\mathcal{P}}$ will never be updated, and the policy will be stuck with a suboptimal decision. Therefore, we need to ensure that the policy queries at least once in every N time slots (for some finite $N \geq 1$), and then use the PSGD approach to update $\hat{\mathcal{P}}$. Finding such an optimal online policy tailored with PSGD updates is a challenging problem, and is a part of our ongoing work. Meanwhile, we may use PSGD with the greedy policy π^g (Algorithm 1), modified to query whenever the last query is N time slots old. We analyze the resulting PSGD+Greedy policy π^{pg} (Algorithm 4) in Section V using numerical simulations.

Algorithm 4 PSGD+Greedy policy π^{pg}.

1: initialize maximum inter-query time $N \geq 1$ (finite);
2: **for** time $t = 1, 2, 3, \cdots$ **do**
3: use π^g (Algorithm 1) with current estimate $\hat{\mathcal{P}}$ to decide whether to predict or query;
4: if query, update $\hat{\mathcal{P}}$ using PSGD (Algorithm 3);
5: **end for**

Remark 5: We use PSGD with π^g instead of $\pi^\star$ (Algorithm 2) since the predict/query decisions in $\pi^\star$ depend intricately on all entries of $\mathcal{P}$, and even small estimation errors can degrade its performance. Moreover, $\pi^\star$ requires running policy iteration (Theorem 2) after each query (i.e., after every $\hat{\mathcal{P}}$ update), making it computationally expensive. In contrast, the PSGD+Greedy policy π^{pg} (Algorithm 4) is significantly simpler to implement.

The following result shows that as time $t \to \infty$, under π^{pg}, $\hat{\mathcal{P}}$ converges to $\mathcal{P}$ (π^{pg} converges to the greedy policy π^g).

Theorem 3: For PSGD, let the learning rate for m^{th} update step be $\eta_m = 1/(8NK+m)$, where N denotes the maximum inter-query time under policy π^{pg} (Algorithm 4) and K is the number of states in $\mathcal{S}$. Under π^{pg}, as $t \to \infty$, the estimate $\hat{\mathcal{P}}$ converges to $\mathcal{P}$, and π^{pg} converges to the greedy policy π^g (Algorithm 1) with probability 1.

Proof Sketch: Since the maximum inter-query time N is finite, as $t \to \infty$, the number of queries (and $\hat{\mathcal{P}}$ updates) made by policy π^{pg} also approaches infinity. The key idea in the proof is to show that as the number of iteration $m \to \infty$, the distance between $\hat{\mathcal{P}}$ and $\mathcal{P}$ converges to zero. The proof involves showing: i) the loss function $\mathcal{F}_{i,n}$ is convex w.r.t. p_{ij}, $\forall\ s_i, s_j \in \mathcal{S}$, and $n \geq 1$, and ii) with convex $\mathcal{F}_{i,n}$ and $\eta_m = 1/(8NK+m)$, each PSGD step ($\hat{\mathcal{P}}$ update) decreases the expected (norm) distance between $\hat{\mathcal{P}}$ and $\mathcal{P}$. ■

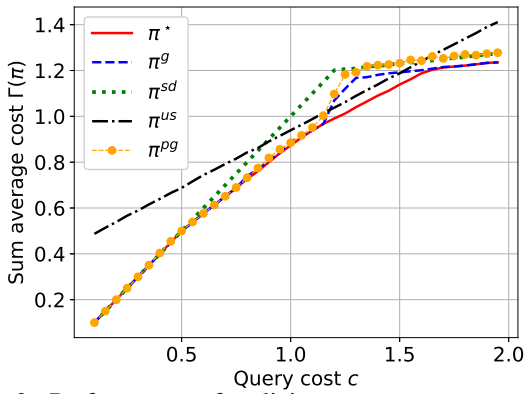

Figure 2: Performance of policies w.r.t. query cost c in a recurrent Markov chain.

V. Numerical Results

Next, we analyze the algorithms and results discussed so far, along with relevant heuristics using numerical simulations.

Remark 6: From Theorem 1, we know that for certain transition probabilities $\mathcal{P}$, the performance of π^g can be arbitrarily bad, essentially due to aggressive prediction. Therefore, in this section, we limit the maximum inter-query time for π^g (and other heuristic policies) to $N = 10$ time slots.

Consider a Markov chain $\mathcal{M}$ with $K = 5$ states, $\mathcal{P} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.6 & 0.2 & 0 \\ 0.2 & 0.1 & 0 & 0.5 & 0.2 \\ 0.1 & 0.2 & 0.2 & 0 & 0.5 \\ 0.1 & 0.1 & 0.2 & 0.3 & 0.3 \end{bmatrix}$, and $\mathcal{L} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}$. For $\mathcal{M}$, we simulate policies $\pi^\star$, π^g and π^{pg} (Algorithm 4), along with two heuristic policies: i) uniform sampling policy π^{us}, which queries at a fixed time interval $\Delta^{us} = 2$ (predicts $k^\star$ (2) otherwise), and ii) a policy π^{sd}, which is similar to π^g, except that for prediction it uses stationary distribution of the Markov chain $\mathcal{M}$ instead of the transition probabilities from the last queried state. Note that for considered $\mathcal{P}$, the stationary distribution $p_{ij}^{(\infty)} = 0.2$, $\forall s_i, s_j \in \mathcal{S}$.

We simulate the policies for different values of query cost c, and compute their sum average cost $\Gamma(\pi)$ by averaging over 100000 time slots. The results are shown in Figure 2. As expected, when the query cost is small, except for π^{us} that queries at fixed interval Δ^{us}, all other policies query more aggressively, and perform identically. Likewise, when the query cost is too large, policies rely mostly on predictions, again leading to similar performance. However, with moderate query cost (i.e., $c \in [1.25, 1.5]$), $\pi^\star$ outperforms all others.

Recall that π^{pg} is expected to converge to π^g. However, in Figure 2, for large c, there is a slight gap in their performance. This is because when c is large, there are fewer queries, and the estimate $\hat{\mathcal{P}}$ is updated less frequently. Therefore, when c is large, convergence requires longer simulations in time.

From Figure 2, we observe that the performance of the uniform sampling policy π^{us} strictly depends on the query cost c, and its performance can be poor if the sampling interval Δ^{us} is not optimized for c. For the above simulation, we optimized

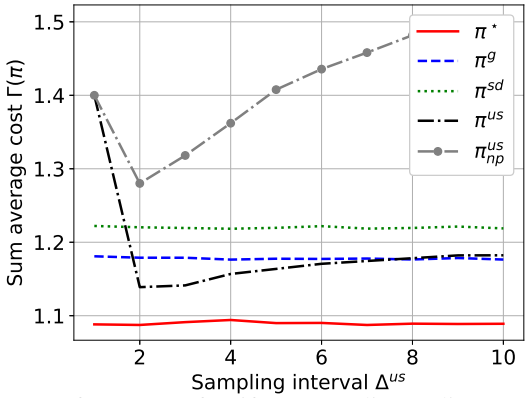

Figure 3: Performance of uniform sampling policy w.r.t. sampling interval, and prediction strategy.

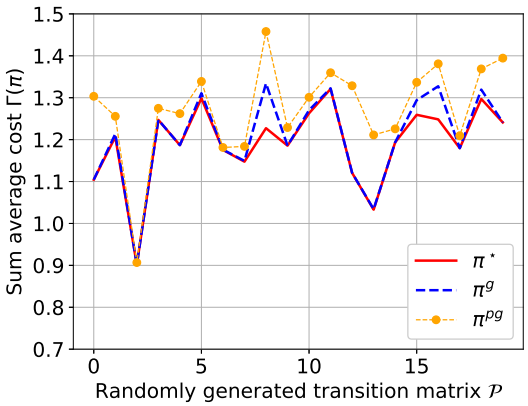

Figure 4: For randomly generated matrices $\mathcal{P}$, the performance of greedy policy π^g is close to the optimal policy $\pi^\star$.

Δ^{us} for $c = 1.4 \in [1.25, 1.5]$ where the performance of the policies differ the most. Figure 3 shows the performance of π^{us} under the same setting (as previous simulation) with $c = 1.4$ for different values of Δ^{us}. We find that for certain values of Δ^{us}, π^{us} outperforms the greedy policy π^g, and π^{sd}. However, π^{us} is never able to match the performance of $\pi^\star$ (Algorithm 2) which has state-dependent sampling intervals.

Note that Figure 3 additionally shows the sum average cost $\Gamma(\pi^{us}_{np})$ for a uniform sampling policy π^{us}_{np}, which is identical to π^{us} except that instead of using the optimal prediction strategy (2), it assumes the next state is identical to the last queried state. Clearly, π^{us}_{np} performs worse among all policies, illustrating the importance of prediction strategy (2).

Finally, to ensure that our observations are not tailored to few specific Markov chains, we simulate the policies for multiple randomly generated transition probability matrices $\mathcal{P}$ (other parameters identical to the previous simulation). Interestingly, we find that in this case, as shown in Figure 4, the performance of the greedy policy π^g is close to $\pi^\star$. This could be attributed to the fact that in each iteration, the elements of matrix $\mathcal{P}$ are identically distributed. Therefore, the uncertainty in the state trajectory from each state is similar.

Thus, comparing the result in Figure 4 with those of previous simulations, we find that $\pi^\star$ is most useful when $\mathcal{P}$ is such that from any state the node is most likely to transition to a small subset of states than the others, i.e., the uncertainty regarding future state is low. Otherwise, π^g is a better choice due to its simplicity and ease of implementation.

VI. Conclusion

For the considered problem pertaining to the query-predict tradeoff, we found that in the Markovian setting, a state-based threshold policy $\pi^\star$ is optimal. However, determining the optimal thresholds is computationally expensive, especially, in dynamic settings where the thresholds need to be updated frequently (e.g. when certain parameters are unknown, and need to be learnt from observations). Therefore, we considered a greedy policy π^g, which is easy to implement, even in non-Markovian/dynamic settings. We saw that despite its simplicity, the greedy policy is near-optimal when the transition probabilities across states are identically distributed, e.g. in settings with limited prior knowledge of the environment or the state transitions. This is particularly relevant in modern systems where state-transitions is often non-Markovian and highly complex to be modeled accurately. However, in setting with absorbing states (or groups of states) π^g must be implemented carefully, with periodic/random queries to improve prediction quality when the node is in one such state.

References

[1] K. Saurav and A. K. R. Chavva, "Minimizing age of information under latency and throughput constraints," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 3676–3681.

[2] J. Niño-Mora, "Markovian restless bandits and index policies: A review," *Mathematics*, vol. 11, no. 7, p. 1639, 2023.

[3] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2731–2735.

[4] K. Saurav and R. Vaze, "Online energy minimization under a peak age of information constraint," *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 579–590, 2023.

[5] B. Joshi, R. V. Bhat, B. Bharath, and R. Vaze, "Minimization of age of incorrect estimates of autoregressive markov processes," in *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 2021, pp. 1–8.

[6] M. Salimnejad, M. Kountouris, A. Ephremides, and N. Pappas, "Age of information versions: a semantic view of markov source monitoring," *IEEE Transactions on Communications*, 2025.

[7] Y. Sun, Y. Polyanskiy, and E. Uysal, "Sampling of the wiener process for remote estimation over a channel with random delay," *IEEE Transactions on Information Theory*, vol. 66, no. 2, pp. 1118–1135, 2019.

[8] T. Z. Ornee and Y. Sun, "Sampling and remote estimation for the ornstein-uhlenbeck process through queues: Age of information and beyond," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 1962–1975, 2021.

[9] T. Z. Ornee, M. K. C. Shisher, C. Kam, and Y. Sun, "Remote safety monitoring: Significance-aware status updating for situational awareness," *arXiv preprint arXiv:2507.09833*, 2025.

[10] Y. Hao, A. Orlitsky, and V. Pichapati, "On learning markov chains," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[11] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

[12] S. Lacoste-Julien, M. Schmidt, and F. Bach, "A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method," *arXiv preprint arXiv:1212.2002*, 2012.