A Learning Approach to Minimum Delay Routing in Stochastic Queueing Networks

Xinzhe Fu and Eytan Modiano
Lab for Information and Decision Systems, MIT

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10229039

***Abstract*—We consider the minimum delay routing problem in stochastic queueing networks where the goal is to find the optimal static routing policy that minimizes the average delay in the network. Previous works on minimum delay routing rely on knowledge of the delay function that maps the routing policies to their corresponding average delay, which is typically unavailable in stochastic queueing networks due to the complex dependency of the delay function on the distributional characteristics of network links. In this paper, we propose a learning approach to the minimum delay routing problem, whereby instead of relying on aprior information on the delay function, we seek to learn the delay function through observations. We design an algorithm that leverages finite-time observations of network queue lengths to approximate the values of the delay function, uses the approximate values to estimate the gradient of the delay function, and performs gradient descent based on the estimated gradient to optimize the routing policy. We prove that our algorithm converges to the optimal static routing policy when the delay function is convex, which is a reasonable condition in practical settings. We conduct extensive simulations to evaluate the empirical performance of our algorithm, demonstrating its superior delay performance over static policies and even dynamic policies such as Join-the-Shortest-Queue and BackPressure.**

I. Introduction

The minimum delay routing problem studies the optimal assignment of traffic between sources and destinations to network paths to minimize the average delay in the network [3], [4]. As delay is one of the most important performance metric in data networks, minimum delay routing has been an important problem in various networking applications such as wireless networks [10], cellular networks [11] and data center networks [9].

The formulation of minimum delay routing was established in the seminal papers by Cantor and Gerla [3] and Gallager [4], where the problem was cast as a non-linear multi-commodity flow problem with the objective function being a *delay function* that maps traffic assignments to the corresponding network delay. Building on this formulation, Gallager proposed an algorithm that computes the optimal assignment based on the incremental delay of paths, which is essentially the gradient of the delay function. Subsequent works on the minimum delay routing problem proposed algorithm using different optimization techniques including first-order methods [5], second-order methods [6], and online approximation [7].

This work was funded by Office of Naval Research (ONR) grant award N00014-20-1-2119, and by NSF grants CNS-2148128 and CNS-2148183.

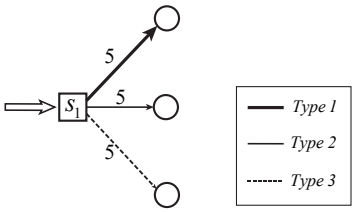


Fig. 1. A Single-hop Network with Links of Different Characteristics.

The central modeling assumption of minimum delay routing is that the functional form of the delay function that characterizes the relationship between the routing assignment and the network delay is known. This might be reasonable for deterministic flow networks, however, in stochastic queueing networks, the relationship between the network delay and routing assignment hinges on not only the service rates but also the distributional characteristics of network links which are often unknown to the network operator. For example, consider the single-hop network shown in Figure 1 with a deterministic arrival process and links of three types. All three links have a mean service rate of 5. The service rate of type 1 link is deterministic, the service rate of type 2 link follows Possion distribution, and the service rate of type 3 link follows a bursty distribution (more details on this example will be presented in Section V). The delay functions of the three links are vastly different and difficult to know in advance, and obviously in more complex settings, characterizing the delay function along paths becomes intractable [8]. Therefore in stochastic queueing networks, previous formulation and results on minimum delay routing are largely inapplicable.

In this paper, we revisit the minimum delay routing problem in stochastic queueing networks. We consider a discrete-time network with K paths between a source and a destination.[1] The number of packets that arrive during each time slot form a a sequence of i.i.d. random variables and the source employs a static (randomized) policy parameterized by a routing vector $\boldsymbol{r} = (r_1, \ldots, r_K)$ with $\sum_{k=1}^{K} r_k = 1$ that sends the arrived packets along path k with probability r_k. Each network link has a queue that buffers the incoming packets. Let $D(\boldsymbol{r})$ be the unknown delay function, defined as the steady-state total queue length of the network under routing vector $\boldsymbol{r}$. The goal is to find the routing vector $\boldsymbol{r}$ that minimizes $D(\boldsymbol{r})$, as by

[1] We present our results in a single-commodity setting for ease of notations. Our results can be easily generalized to networks with multiple sources and multiple destinations.

Little's law, minimizing $D(\boldsymbol{r})$ is equivalent to minimizing the steady-state delay.

Note that our goal is to identify the optimal static routing policy, where the routing decisions are independent of queue lengths. This is in contrast with a recent line of work that aims to design optimal dynamic routing policies, where the routing decisions can be made based on queue lengths, in stochastic queueing networks [15], [16]. As dynamic routing policies are more general than static routing policies, the optimal dynamic routing policy can achieve a smaller steady-state delay than the optimal static routing policy. However, the optimal static policy has two main advantages over its dynamic counterpart. First, static policies are easier to implement and more widely deployed in real systems, which makes them more relevant to network optimization in practice [9], [10]. Second, more importantly, since the problem of finding the optimal dynamic policy that minimizes steady-state queue length is a Markov-decision process with infinite state space [21], few results are known except for some special cases such as Join-the-Shortest-Queue for symmetric single-hop networks [12] and $c\mu$-rule for homogeneous single-hop networks [13], [14]. Due to the inherent complexity of the problem, previously proposed dynamic policies for general networks [15], [16], [19], [20] can only be shown to stabilize the network but have no guarantee on delay performance. It is also worth noting that in many cases dynamic policies have worse delay performance than the optimal static policy. Again, consider the simple network in Figure 1, when the arrival process is deterministic with rate smaller than 5, the optimal static routing policy is to send all the packets along the deterministic link (type 1), incurring no delay. In contrast, dynamic policies such as Joint-the-Shortest-Queue, would unavoidably send some packets to the Poisson link and the bursty link (types 2 and 3) and incur a higher delay, which is verified by our simulation results (see Section V).

The key challenge in finding the optimal static routing policy lies in the unknown delay function $D(\boldsymbol{r})$. We tackle the challenge by leveraging the following insight: for a routing vector $\boldsymbol{r}$, if we operate the network under $\boldsymbol{r}$, the total qeueue length will converge to $D(\boldsymbol{r})$ in expectation. Therefore, the queue length observation after operating the network under $\boldsymbol{r}$ for a sufficient period of time can serve as a stochastic approximation to $D(\boldsymbol{r})$. Such queue length observations enable us to essentially *learn* D and construct anestimator of the gradients of D following the recently proposed framework of learning in stochastic queueing networks [27], [26]. Based on this idea, we propose the *gradient sampling* algorithm for the minimum delay routing problem in stochastic queueing networks, where we learn the values of $D(\boldsymbol{r})$ through sampling the queue lengths, use the values to construct approximate gradients of $\nabla D(\boldsymbol{r})$, and find the optimal $\boldsymbol{r}$ via gradient descent. For example, in the network in Figure 1 with an arrival rate no larger than 5, through queue length observations, our algorithm can approximately learn that the delay function D does not increase if we send more packets on the deterministic link, and eventually converge to the optimal policy that sends all the packets along the deterministic link.

Leveraging the convergence bounds of countable-state markov chains [29], we show that when the delay function $D(\boldsymbol{r})$ is convex, the gradient sampling algorithm converges to the optimal routing vector as it achieves a sublinear regret of $O(T^{3/4}\ln^2 T)$ over a time horizon of T time slots. Furthermore, we prove that when the network paths are non-overlapping, the function $D(\boldsymbol{r})$ is indeed convex, which shows that the gradient sampling algorithm can find the optimal static routing policy for networks with non-overlapping paths. Note that as previously the convexity of $D(\boldsymbol{r})$ has only been established for single-queues [30] or single-hop networks [31], our convexity results are of independent interest. Finally, we evaluate the empirical performance of the gradient sampling algorithm through simulations on networks with parallel links, parallel paths, and general topology(with overlapping paths). Our simulation results validate the efficacy of the gradient sampling algorithm and show its superiority in terms of delay minimization even over established dynamic policies such as Join-the-Shortest-Queue, Back-Pressure [35], and Universal Max-Weight [17].

The rest of the paper is organized as follows. In Section II, we introduce the model and problem formulation of the minimum delay routing problem in stochastic queueing networks. In Section III, we present the gradient sampling algorithm. We analyze the theoretical performance of gradient sampling in Section IV, and then evaluate its empirical performance via simulations in Section V. Finally, we conclude the paper in Section VI.

II. Model and Problem Formulation

A. Network Model

Consider a network $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $\mathcal{V}$ representing the set of nodes and $\mathcal{E}$ representing the set of links. For simplicity of presentation, we assume that there is a single source $s \in \mathcal{V}$ and a singe destination $d \in \mathcal{V}$ in the network. Extension to the multi-source/multi-destination setting is straightforward. The traffic can be routed on K paths $\{\mathcal{P}_1,\ldots,\mathcal{P}_K\}$ from s to d. For a link $e \in E$, we write $e \in \mathcal{P}_k$ if link e is on path $\mathcal{P}_k$.

The network operates in discrete time. At every time slot t, $a(t)$ packets arrive at s, with $a(t)$ being a sequence of i.i.d. bounded random variables with $\mathbb{E}[a(t)] = \lambda$ and $|a(t)| \le C_0$. The source employs a randomized routing policy that is parameterized by a *routing vector* $\boldsymbol{r} = (r_1,\ldots,r_K)$ with $\sum_{k=1}^{K} r_k = 1, r_k \ge 0$, where at every time slot, the arrived packets are routed on path $\mathcal{P}_k$ with probability r_k.

At every link e, there is a queue Q_e that buffers the packets coming into link e. At each time t, the offered service of link e is denoted as $c_e(t)$ with $c_e(t)$ being a sequence of i.i.d. bounded random variables with $\mathbb{E}[c_e(t)] = \mu_e$ and $|c_e(t)| \le C_0$. The links employ the first-in-first-out service discipline. Let $Q_e(t)$ be the queue length of link e at time t, and $a_e(t)$ be the total number of packets that arrived at link e at t, then

the queue length dynamics follow the recursion $Q_e(t+1) := [Q_e(t) + a_e(t) - c_e(t)]^+$.[2]

B. Problem Formulation

Define the stability region as the set of routing vectors $\boldsymbol{r}$ under which the network queues are stable. More specifically, define $\Lambda := \{\boldsymbol{r} \mid \forall e \in \mathcal{E}, \sum_{k:e\in\mathcal{P}_k} \lambda r_k \leq \mu_e\}$. From the result of [1], [2], when the routing vector is in the interior of Λ, i.e., $\boldsymbol{r} \in int(\Lambda)$, we have that the queues are stable and the markov chain formed by the queues is positive recurrent. Therefore, there exists a unique stationary distribution of the queue lengths $\{Q_e, e \in \mathcal{E}\}$. For each routing vector $\boldsymbol{r} \in int(\Lambda)$, we define the stationary distribution as $\pi_{\boldsymbol{r}}$. By standard results in Markov-chain theory [29], as t goes to infinity, $\{Q_e(t), e \in \mathcal{E}\}$ converges to $\pi_{\boldsymbol{r}}$ in distribution, and $\lim_{t\to\infty} \mathbb{E}[\sum_{e\in\mathcal{E}} Q_e(t)] = \mathbb{E}_{\pi_{\boldsymbol{r}}}[\sum_{e\in\mathcal{E}} Q_e]$. By Little's law, the average steady-state delay experienced by the packets is equal to $\mathbb{E}_{\pi_{\boldsymbol{r}}}[\sum_{e\in\mathcal{E}} Q_e]/\lambda$. Therefore, minimizing the steady-state delay is equivalent to minimizing the steady-state total queue length. We define the delay function $D(\boldsymbol{r}) := \mathbb{E}_{\pi_{\boldsymbol{r}}}[\sum_{e\in\mathcal{E}} Q_e]$. The minimum delay routing problem in stochastic queueing networks can thus be formulated as seeking the routing vector in $int(\Lambda)$ that minimizes the delay function $D(\boldsymbol{r})$.

Definition 1 (Minimum Delay Routing)**.** *The minimum delay routing problem seeks for the routing vector* $\boldsymbol{r}^* \in int(\Lambda)$ *that minimizes the delay function* $D(\boldsymbol{r})$*, i.e.,* $\boldsymbol{r}^* := \arg\min_{\boldsymbol{r}\in int(\Lambda)} D(\boldsymbol{r})$.

When $\boldsymbol{r}$ is on the boundary of Λ or outside Λ, the queues are not stable and steady state distribution does not exist. We can alternatively extend the definition of the delay function to this case by setting $D(\boldsymbol{r})$ as $+\infty$ when $\boldsymbol{r} \notin int(\Lambda)$ and remove the constraint $\boldsymbol{r} \in int(\Lambda)$ in the formulation.

Note that in this work, we focus on optimizing over the set of static randomized policies and do not consider dynamic policies that can depend on queue lengths. Therefore, there exist cases where some dynamic policies like Join-the-Shortest-Queue can achieve a smaller steady-state delay than $\boldsymbol{r}^*$. However, static policies are of great significance since they are widely implemented in practice due to their simplicity, and as we will show, in many settings the optimal static policy can outperform established dynamic policies, since the latter offer no guarantee on delay performance.

III. The Gradient Sampling Algorithm

In this section, we propose the *gradient sampling* algorithm for the minimum delay routing problem in stochastic queueing networks. We will first give a high-level overview and then present the details of the policy.

[2]Note that queue, arrived packets and departed packets at link e may contain packets on every path k with $e \in \mathcal{P}_k$. We do not keep track of the number of packets on each path in the queues since they are not necessary for our analysis and would greatly complicate the notations.

A. Overview

The minimum delay routing problem solves the following optimization problem $\mathcal{P}$ in the context of stochastic queueing networks.

$$(\mathcal{P}) \textbf{ Minimize: } D(\boldsymbol{r}) \tag{1}$$
$$\textbf{s.t. } \boldsymbol{r} \in \Lambda \tag{2}$$
$$\sum_{k=1}^{K} r_k = 1, \tag{3}$$
$$r_k \geq 0, \ \forall k. \tag{4}$$

A natural idea for solving $\mathcal{P}$ is the projected gradient descent method that has been explored in [4], [22]. Let $\nabla D(\boldsymbol{r})$ be the gradient of D at $\boldsymbol{r}$, and $\Lambda_{\mathcal{P}} := \{\boldsymbol{r} \mid \boldsymbol{r} \in \Lambda, \sum_{k=1}^{K} r_k = 1, r_k \geq 0, \ \forall k\}$ be the feasibility region of $\mathcal{P}$. The projected gradient descent method maintains a routing vector $\boldsymbol{r}_t$ at each time t and performs the update $\boldsymbol{r}_{t+1} := \Pi_{\Lambda_{\mathcal{P}}}[\boldsymbol{r}_t - \eta_t \nabla D(\boldsymbol{r}_t)]$, where $\eta_t > 0$ is the step-size at time t and $\Pi_{\Lambda_{\mathcal{P}}}$ represents the projection onto the set $\Lambda_{\mathcal{P}}$.

In stochastic queueing networks, the objective function D is unknown. Thus, the projected gradient descent is not applicable since we cannot evaluate the gradient $\nabla D(\boldsymbol{r})$. In proposing the gradient sampling algorithm, we leverage two ideas to deal with the unknown delay function under the projected gradient descent framework. First, we use the gradient estimator proposed in [24], [26] to convert the task of evaluating the gradients of D to evaluating the function values of D. More specifically, consider a random perturbation vector $\boldsymbol{\epsilon}$. The result in [24] shows that when $\boldsymbol{\epsilon}$ is sampled appropriately, in expectation, the quantity $\frac{K[D(\boldsymbol{r}+\boldsymbol{\epsilon})-D(\boldsymbol{r}-\boldsymbol{\epsilon})]\cdot\boldsymbol{\epsilon}}{2||\boldsymbol{\epsilon}||}$, which is similar to a finite-difference estimator, is close to $\nabla D(\boldsymbol{r})$. Therefore, if we can evaluate the values of $D(\boldsymbol{r}+\boldsymbol{\epsilon})$ and $D(\boldsymbol{r}-\boldsymbol{\epsilon})$, we will have an approximation of $\nabla D(\boldsymbol{r})$. Second, for a routing vector $\boldsymbol{r}$, if we employ the static routing policy parameterized by $\boldsymbol{r}$ for a sufficiently long period, then the expected total queue length should converge to $D(\boldsymbol{r})$. Therefore, we can approximate the values of $D(\boldsymbol{r}+\boldsymbol{\epsilon})$ and $D(\boldsymbol{r}-\boldsymbol{\epsilon})$ by operating the network under $\boldsymbol{r}+\boldsymbol{\epsilon}$ and $\boldsymbol{r}-\boldsymbol{\epsilon}$ for sufficiently long and taking the observed total queue lengths, respectively. The gradient sampling algorithm uses this idea combined with the gradient estimator $\frac{K[D(\boldsymbol{r}+\boldsymbol{\epsilon})-D(\boldsymbol{r}-\boldsymbol{\epsilon})]\cdot\boldsymbol{\epsilon}}{2||\boldsymbol{\epsilon}||}$ to find the optimal routing vector $\boldsymbol{r}^*$ following the framework of projected gradient descent.

Let $\mathcal{B}_K$ be the intersection of K-dimensional unit sphere and the hyper-plane $\{\boldsymbol{x} \in \mathbb{R}^K \mid \sum_{k=1}^{K} x_k = 0\}$. We will construct the perturbation $\boldsymbol{\epsilon}$ by sampling $\boldsymbol{u}$ uniformly from $\mathcal{B}_K$ and use $\delta\boldsymbol{u}$ as the perturbation vector where $\delta > 0$ is a small scalar. We sample the vector from te intersection of the unit sphere and the hyper-plane to maintain that the components of the routing vectors $\boldsymbol{r} \pm \delta\boldsymbol{u}$ sum up to one.

B. Details of Gradient Sampling

Now, we present the details of our gradient sampling algorithm. The pseudo-code is shown in **Algorithm 1**. In Algorithm 1, the set $\Lambda_\epsilon \subseteq \Lambda_{\mathcal{P}}$ is defined as the set of routing

vectors that are at least ϵ-away in the Euclidean norm from the boundary of $\Lambda_{\mathcal{P}}$. The gradient sampling algorithm proceeds in an episodic fashion, with each episode having 2τ time slots. It updates the maintained routing vector every episode. For each episode i, let t_i be the index of the first time slot of episode i, and $\boldsymbol{r}_i$ be the maintained routing vector at episode i. The policy first samples a vector $\boldsymbol{u}$ uniformly from the set $\mathcal{B}_K$. Next, it employs the routing vector $\boldsymbol{r}_i + \delta\boldsymbol{u}$ for τ time slots. For this period of τ time slots, the first $\tau - 1$ slots serve as a burn-in period for the network to converge to the steady-state under the routing vector, and the queue lengths at the τ-th slot are taken as an approximation to the steady-state queue lengths under routing vector $\boldsymbol{r}_i+\delta\boldsymbol{u}$ (Line 6). Then, the policy employs the routing vector $\boldsymbol{r} - \delta\boldsymbol{u}$ for τ time slots, and again sums the system queue length at the last time slot as $\hat{D}(\boldsymbol{r}_i + \delta\boldsymbol{u})$ (Line 8). The approximate queue lengths under $\boldsymbol{r}_i + \delta\boldsymbol{u}$ and $\boldsymbol{r}_i - \delta\boldsymbol{u}$ are then used to construct an approximate gradient $\hat{\nabla}D(\boldsymbol{r}_i)$ of D at $\boldsymbol{r}_i$ (Line 9). Finally, the policy updates the routing vector $\boldsymbol{r}_i$ via a step of projected gradient descent using the approximate gradient $\hat{\nabla}D(\boldsymbol{r}_i)$. The projection is done onto the set Λ_ϵ with $\epsilon > \delta$, which ensures that the routing vectors $\boldsymbol{r}_i - \delta\boldsymbol{u}$ and $\boldsymbol{r}_i + \delta\boldsymbol{u}$ lie in the stability region.

Algorithm 1 The gradient sampling algorithm

Require: Network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, parameters $\tau, \delta, \eta, \epsilon$. $\epsilon > \delta$.

1: **Initialize:** $\boldsymbol{r}_0 \in \Lambda_\epsilon$.
2: **for** episode $i = 1, \ldots,$ **do**
3: $t_i :=$ the first time slot of episode i.
4: Sample vector $\boldsymbol{u}$ uniformly from $\mathcal{B}_K$.
5: Use routing vector $\boldsymbol{r}_i + \delta\boldsymbol{u}$ for τ time slots.
6: $\hat{D}(\boldsymbol{r}_i + \delta\boldsymbol{u}) := \sum_{e\in\mathcal{E}} Q_e(t + \tau - 1)$.
7: Use routing vector $\boldsymbol{r}_i - \delta\boldsymbol{u}$ for τ time slots.
8: $\hat{D}(\boldsymbol{r}_i + \delta\boldsymbol{u}) := \sum_{e\in\mathcal{E}} Q_e(t + 2\tau - 1)$.
9: $\hat{\nabla}D(\boldsymbol{r}_i) := \frac{K[\hat{D}(\boldsymbol{r}_i+\delta\boldsymbol{u}) - \hat{D}(\boldsymbol{r}_i-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}$.
10: $\boldsymbol{r}_{i+1} := \Pi_{\Lambda_\epsilon}[\boldsymbol{r}_i - \eta\hat{\nabla}D(\boldsymbol{r}_i)]$.

1) Choice of Parameters: **Algorithm 1** involves four input parameters τ, δ, η and ϵ. We now comment on their roles in the gradient sampling algorithm and how their values affect the performance of the policy.

- The parameter τ governs the number of samples we use to approximate the steady state queue lengths. A larger value of τ makes our approximations closer to the true value of steady state queue lengths (Lines 6 and 8) which will lead to a more accurate approximate gradient, but on the other hand it reduces the number of updates completed in a given time horizon.
- The parameter δ controls the magnitude of the perturbations vectors we use to approximate the gradients. Theoretically, if we have access to the exact values of steady state queue lengths $D(\boldsymbol{r} + \delta\boldsymbol{u})$ and $D(\boldsymbol{r} - \delta\boldsymbol{u})$, the error of the ideal gradient estimator $\frac{K[D(\boldsymbol{r}+\delta\boldsymbol{u}) - D(\boldsymbol{r}-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}$ decreases with δ. However, as we are using finite-time observations to approximate the steady state queue lengths, the randomness of the finite-time observations will make the variance of the approximate gradient $\hat{\nabla}D(\boldsymbol{r})$ grow as δ decreases. Therefore, when choosing the value of δ we need to consider the trade-off between the bias and variance of the approximate gradients.
- The parameter η sets the step size of the projected gradient descent. We need to appropriately choose its value to match the convergence rate of the policy.
- The parameter ϵ essentially restricts the set of routing vectors we optimize over. As we will see in the theoretical analysis, a large value of ϵ makes the algorithm converges faster since the queue lengths tend to be small during the course of the optimization as the routing vectors are further away from the boundary of the capacity region. On the other hand, we need to make sure that the optimal routing vector $\boldsymbol{r}^* \in \Lambda_\epsilon$, which means that ϵ cannot be too large, especially when the network is heavily-loaded.

Remark: There are several variations of the gradient sampling algorithm alternative to the one presented in Algorithm 1. First, in contrast with sampling $\boldsymbol{u}$ from $\mathcal{B}_K$ (which we adopt due to its simplicity in presentation and analysis), we can use other sampling procedures. For example, randomly choose $k, k' \in \{1, \ldots, K\}$, and set $\boldsymbol{u}$ as a vector with the k-th component being 1, the k'-th component being -1, and other components being 0. Our theoretical and empirical results also apply to this sampling procedure. Second, instead of having the parameters being fixed, the values of the parameters can change with time. The policy presented in Algorithm 1 where the parameter values are fixed ahead of time is easier to implement. It is suitable for the applications where the gradient sampling algorithm is used to identify the optimal static policy and the network will operate under the routing vector found by the gradient sampling algorithm, which can be taken as $\boldsymbol{r}_i$ of the last iteration or the average of $\boldsymbol{r}_i$ over all the completed iterations. Alternatively, if we want to directly run the gradient sampling algorithm as the operating routing policy of the network, we can set δ, η to be decreasing with time such that eventually $\boldsymbol{r}_i + \delta\boldsymbol{u}$ and $\boldsymbol{r}_i - \delta\boldsymbol{u}$ converge to the optimal static policy.

IV. Theoretical Analysis

In this section, we analyze the theoretical performance of the gradient sampling algorithm. We will show that when the delay function D is convex and Lipschitz-continuous, the gradient sampling algorithm converges to the optimal routing vector $\boldsymbol{r}^*$ and present bounds on the convergence rate.

A. Condition on the Delay Function

We will show that the gradient sampling algorithm converges to the optimal routing vector under the following Condition.

Condition 1: $D(\boldsymbol{r})$ is a L-Lipschitz[3] and convex function of $\boldsymbol{r}$.

The Lipschitz continuity of D is easy to establish since D is a bounded continuous function on a bounded domain. For the

[3] D is L-Lipschitz if $|D(\boldsymbol{r}_1) - D(\boldsymbol{r}_2)| \leq L||\boldsymbol{r}_1 - \boldsymbol{r}_2||_2$ where $||\cdot||_2$ denotes the Euclidean norm.

convexity, previous works have shown that the delay function D is indeed convex in single-hop networks, i.e., where each path $\mathcal{P}_k$ consists of a single queue [32], [31]. In this paper, we prove a more general result that D is also convex for networks consisting of non-overlapping paths, i.e., there does not exist a link $e \in \mathcal{E}$ that lies on more than one paths.

Proposition 1. *In networks where $\{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$ do not overlap, the delay function D is a convex function of $\boldsymbol{r}$.*

Proposition 1 is proved using the theory of stochastic coupling and stochastic convexity [33]. Due to the space limits, we omit the details of the proof. Note that Proposition 1 shows that Condition 1 holds for networks with non-overlapping paths. Based on simulation results, we conjecture that Condition 1 still holds in general networks, however, the analysis of networks with overlapping paths is much more complicated and left for future work.

B. Preliminaries

Next, we proceed to prove the convergence of the gradient sampling algorithm. We start by presenting preliminary results that will be useful in establishing the convergence of the gradient sampling algorithm. In the proceeding analysis, we will often encounter constants that do not grow with the time horizon T. For simplicity of notations, we will use C to represent all the constants whenever convenient. The constants C in different equations may not be the same. As we will make use of Lyapunov theory in the analysis, we first introduce two auxiliary lemmas related to the existence and properties of Lyapunov functions for the Markov chain induced by the network queues.

The first lemma comes from [28], which shows the existence of a Lyapunov function of the queue length vector that has negative drift for all routing vectors $\boldsymbol{r} \in \Lambda_\epsilon$

Lemma 1 (Theorem in [28])**.** *Let $\boldsymbol{Q}(t)$ be the vector of queue lengths at time t. There exists an Lyapunov function $f(\boldsymbol{Q}(t))$ and a constant C such that for any $\boldsymbol{r} \in \Lambda_\epsilon$, $\mathbb{E}[f(\boldsymbol{Q}(t+1)) - f(\boldsymbol{Q}(t)) \mid f(\boldsymbol{Q}(t)) \geq C] \leq -\epsilon$. Furthermore, there exists constants $C_2 > C_1 > 0$ with $C_1 \sum_{e\in\mathcal{E}} Q_e(t) \leq f(\boldsymbol{Q}(t)) \leq C_2 \sum_{e\in\mathcal{E}} Q_e(t)$.*

Remark: It was shown in [28] that the Lyapunov function f can be constructed as a piece-wise linear function via solving a linear program, and the constants C_0, C_1, C_2 depend on the network topology. Note that our analysis only relies on the existence of such Lyapunov function but does not need to know the Lypunov function.

The second lemma comes from [29]. It establishes a bound on the convergence rate of countable-state Markov chains to the steady-state based on the existence of a Lyapunov function.

Lemma 2 (Theorem 7.1.2 in [29])**.** *Consider a countable-state Markov chain with state space $\mathcal{S}$ and transition kernel $\{p_{ij}\}$, i.e., for $i, j \in \mathcal{S}$, p_{ij} is the one-step transition probability from state i to state j. If there exists a Lyapunov function $f : \mathcal{S} \mapsto \mathbb{R}^+$ that satisfies the following conditions:*

1) *There exists a constants $C, \epsilon > 0$ such that for all $i \in \mathcal{S}$ with $f(i) \geq C$, $\sum_{j\in\mathcal{S}} p_{ij} f(j) - f(i) \leq -\epsilon$.*
2) *There exists a constant C such that for any $i, j \in \mathcal{S}$ with $p_{ij} > 0$, $|f(i) - f(j)| \leq C$,*

then the Markov chain is positive-recurrent with steady state distribution π that satisfies:

(a) There exists $C, a_1 > 0$ such that $\pi(i) < C\exp(-a_1 f_i)$ for all $i \in \mathcal{S}$.
(b) There exists constants $C, a_2 > 0$ such that $\sum_{j\in\mathcal{S}} |p_{ij}^{(n)} - \pi_j| < C\exp(-a_2 n)$ for all $i \in \mathcal{S}, n > 2f(i)$,

where $p_{ij}^{(n)}$ is the n-step transition probability from state i to state j.

Based on Lemmas 1 and 2, we establish the following proposition that bounds the error of $\hat{D}$ used in the gradient sampling algorithm (Lines 6 and 8). In the proposition, we consider a generic routing vector $\boldsymbol{r} \in \Lambda_\epsilon$, and let f be the Lyapunov function for the network defined in Lemma 1.

Proposition 2. *Starting from any state $\boldsymbol{Q}(t)$ at time t, for any $\tau \geq 2f(\boldsymbol{Q}(t))$, there exists constants $C, a_3 > 0$ such that $\mathbb{E}|\hat{D}(\boldsymbol{r}) - D(\boldsymbol{r})| \leq C\exp(-a_3\tau)$.*

Proof. By Lemma 1, there exists a Lyapunov function f for the Markov chain of queue lengths such that $\mathbb{E}[f(\boldsymbol{Q}(t+1)) - f(\boldsymbol{Q}(t)) \mid f(\boldsymbol{Q}(t)) \geq C_0] \leq -\epsilon$ for some $C_0 > 0$. Also, since the arrivals and services to the queues are bounded and there exist $C_1, C_2 > 0$ with $C_1 \sum_{e\in\mathcal{E}} Q_e(t) \leq f(\boldsymbol{Q}(t)) \leq C_2 \sum_{e\in\mathcal{E}} Q_e(t)$, we have that there exists C such that $|f(\boldsymbol{Q}(t+1)) - f(\boldsymbol{Q}(t))| \leq C$. Therefore, the conditions of Lemma 2 are met. Let $\pi_{\boldsymbol{r}}(\boldsymbol{Q})$ be the probability of state $\boldsymbol{Q}$ in the steady-state queue length distribution and $p_{\boldsymbol{Q}\boldsymbol{Q}'}^{(\tau)}$ be the τ-step transition probability from $\boldsymbol{Q}$ to $\boldsymbol{Q}'$. It follows from Lemma 2(a) that for any queue length state $\boldsymbol{Q}$, there exist $C, a_1 > 0$ such that

$$\pi_{\boldsymbol{r}}(\boldsymbol{Q}) < C\exp(-a_1 f(\boldsymbol{Q})). \tag{5}$$

Also from Lemma 2(b), we have that there exist $C, a_2 > 0$ such that for $\tau \geq 2f(\boldsymbol{Q})$,

$$\sum_{\boldsymbol{Q}'} |p_{\boldsymbol{Q}\boldsymbol{Q}'}^{(\tau)} - \pi_{\boldsymbol{r}}(\boldsymbol{Q}')| > C\exp(-a_2\tau). \tag{6}$$

With slight abuse of notation, we will use j to denote the set of queue length states such that $\sum_{e\in\mathcal{E}} Q_e = j$, and $p_{\boldsymbol{Q}j}^{(n)}$ to denote the n-step transition probabilities from state $\boldsymbol{Q}$ to the set j. Consider any $\tau \geq 2f(\boldsymbol{Q}(t))$. As the arrivals and services are bounded by C_0, after τ time slots, $\sum_{e\in\mathcal{E}} Q_e(t+\tau) \leq \sum_{e\in\mathcal{E}} Q_e(t) + C_0 \cdot |\mathcal{E}|\tau$. Recall from Lemma 1 that τ is lower bounded by $2f(\boldsymbol{Q}(t)) \geq 2C_1 \sum_{e\in\mathcal{E}} Q_e(t)$ for some $C_1 > 0$. It follows that there exists C such that $\sum_{e\in\mathcal{E}} Q_e(t+\tau) \leq C\tau$. Hence, the τ-step transition probability from $\boldsymbol{Q}(t)$ to any set $j > C\tau$ is zero. Thus, we have starting from $\boldsymbol{Q}(t)$ (which will be abbreviated as $\boldsymbol{Q}$ in the following equations), for $\tau \geq 2f(\boldsymbol{Q}(t))$,

$$\mathbb{E}[\hat{D}(\boldsymbol{r}) - D(\boldsymbol{r})]$$

$$=\mathbb{E}|\sum_{e\in\mathcal{E}} Q_e(t+\tau) - D(\boldsymbol{r})| = |\sum_{j=1}^{\infty} p_{\boldsymbol{Q}j}^{(\tau)}\cdot j - \sum_{j=1}^{\infty}\pi_{\boldsymbol{r}}(j)\cdot j|$$

$$\leq \sum_{j=1}^{\infty} |p_{\boldsymbol{Q}j}^{(\tau)} - \pi_{\boldsymbol{r}}(j)|\cdot j \tag{7}$$

$$= \sum_{j=1}^{C\tau} |p_{\boldsymbol{Q}j}^{(\tau)} - \pi_{\boldsymbol{r}}(j)|\cdot j + \sum_{j=C\tau+1}^{\infty} |p_{\boldsymbol{Q}j}^{(n)} - \pi_{\boldsymbol{r}}(j)|\cdot j \tag{8}$$

$$= \sum_{j=1}^{C\tau} |p_{\boldsymbol{Q}j}^{(\tau)} - \pi_{\boldsymbol{r}}(j)|\cdot j + \sum_{j=C\tau+1}^{\infty} \pi_{\boldsymbol{r}}(j)\cdot j \tag{9}$$

$$\sum_{j=1}^{C\tau} |p_{\boldsymbol{Q}j}^{(\tau)} - \pi_{\boldsymbol{r}}(j)|\cdot j + \sum_{j=C\tau+1}^{\infty} \pi_{\boldsymbol{r}}(j)\cdot j \tag{10}$$

$$\leq \sum_{j=1}^{C\tau} C\exp(-a_2\tau)\cdot j + \sum_{j=C\tau+1}^{\infty} C\exp(-a_1 j)\cdot j \tag{11}$$

$$\leq C\exp(-a_3\tau), \tag{12}$$

where (11) follows from (5) and (6), and (12) follows from the telescoping of exponential series. Therefore,

$$\mathbb{E}|\hat{D}(\boldsymbol{r}) - D(\boldsymbol{r})|$$
$$=\mathbb{E}|\sum_{e\in\mathcal{E}} Q_e(t+\tau) - D(\boldsymbol{r})| \leq C\exp(-a_3\tau),$$

which concludes the proof. □

Finally, we introduce a result from [24] showing that $\mathbb{E}_{\boldsymbol{u}\in\mathcal{B}_K}\frac{K[D(\boldsymbol{r}+\delta\boldsymbol{u})-D(\boldsymbol{r}-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}$ is the gradient of an approximate version of D.

Lemma 3 (Lemma 1 in [24])**.** *There exists a convex function F with $|G(\boldsymbol{r})-D(\boldsymbol{r})|\leq C\delta$ for all $\boldsymbol{r}\in\Lambda$ such that $\nabla G(\boldsymbol{r}) = \mathbb{E}_{\boldsymbol{u}\in\mathcal{B}_K}\frac{K[D(\boldsymbol{r}+\delta\boldsymbol{u})-D(\boldsymbol{r}-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}$.*

C. Convergence of Gradient Sampling

We measure the convergence rate of the gradient sampling algorithm with *regret*. Let $\boldsymbol{r}_t$ be the routing vector employed by the gradient sampling algorithm at time t. For a finite time horizon T, the regret is defined as $\sum_{t=1}^{T}\mathbb{E}[D(\boldsymbol{r}_t) - D(\boldsymbol{r}^*)]$. We will show in Theorem 1 that the regret of the gradient sampling algorithm grows sublinearly with T under Condition 1.

Theorem 1. *Under condition 1, the gradient sampling algorithm with $\delta = \frac{1}{T^{1/4}}$, $\eta = \frac{\epsilon}{T^{3/4}}$, $\tau = \frac{\ln^2 T}{\epsilon}$ and ϵ such that $\boldsymbol{r}^*\in\Lambda_\epsilon$ has regret $\sum_{t=1}^{T}\mathbb{E}[D(\boldsymbol{r}_t) - D(\boldsymbol{r}^*)] = O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right)$.*

Theorem 1 has the following implications on the performance of the gradient sampling algorithm. First, as under condition 1, D is convex. By Jensen's inequality, we have $\frac{\sum_{t=1}^{T} D(\boldsymbol{r}_t)}{T} \geq D\left(\frac{\sum_{t=1}^{T}\boldsymbol{r}_t}{T}\right)$. Hence, a sublinear regret bound implies that $D\left(\frac{\sum_{t=1}^{T}\boldsymbol{r}_t}{T}\right) - D(\boldsymbol{r}^*)$ goes to zero as T goes to infinity. which means that if we use the gradient sampling algorithm to find the optimal routing vector, by averaging the routing vectors $\boldsymbol{r}_t$ over the time horizon, our gap to the optimal can be made arbitrarily small. Furthermore, the same technique in [23] can be used to derive from Theorem 1 that $\mathbb{E}[D(\boldsymbol{r}_T) - D(\boldsymbol{r}^*)] = O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right)$. Therefore, the routing vector of the gradient sampling algorithm at the end of the time horizon also converges to the optimal. Second, if we use the gradient sampling algorithm as the operating routing policy for the network, then the regret bound implies that the cumulative gap between the routing vector used by the policy and the optimal grows sublinearly with the time horizon T and thus the time-average gap goes to zero.

The regret bound is summarized in Theorem 1

The key step in proving Theorem 1 is bounding the error induced by the approximate gradient $\hat{\nabla}D(\boldsymbol{r})$. We will use Proposition 2 and Lemma 3 to bound the error of the gradient estimator $\hat{\nabla}D(\boldsymbol{r})$ used in the gradient sampling algorithm with respect to $\nabla\tilde{D}(\boldsymbol{r})$. Plugging that in the standard analysis of projected gradient descent [25], we prove the regret of gradient sampling with respect to $\tilde{D}$. Finally, from the closeness between $\tilde{D}$ and D, we prove the regret of the gradient sampling algorithm with respect to D.

1) Proof of Theorem 1: For ease of presentation, in the proof we will not write out the explicit constants but instead make use of the big-O notation. To begin with, since each episode last for 2τ time slots, there are $\frac{T}{2\tau}$ (assumed to be an integer) episodes over the time horizon of T time slots. Note that during episode i, the gradient sampling uses routing vector $\boldsymbol{r}_i+\delta\boldsymbol{u}$ for τ time slots and routing vector $\boldsymbol{r}_i-\delta\boldsymbol{u}$ for τ time slots. The regret of gradient sampling is thus equal to

$$\sum_{i=1}^{T/(2\tau)} \tau\left(\mathbb{E}[D(\boldsymbol{r}_i+\delta\boldsymbol{u}) - D(\boldsymbol{r}^*)] + \mathbb{E}[D(\boldsymbol{r}_i-\delta\boldsymbol{u}) - D(\boldsymbol{r}^*)]\right) \tag{13}$$

Since by Condition 1, D is Lipschitz continuous on Λ_ϵ, we have $|D(\boldsymbol{r}_i)-D(\boldsymbol{r}_i+\delta\boldsymbol{u})| = O(\delta)$ and $|D(\boldsymbol{r}_i)-D(\boldsymbol{r}_i-\delta\boldsymbol{u})| = O(\delta)$. As $\delta = \frac{1}{T^{1/4}}$, $T\delta = T^{3/4}$. It follows that the term

$$\sum_{i=1}^{T/(2\tau)} \tau\left(\mathbb{E}[D(\boldsymbol{r}_i+\delta\boldsymbol{u}) - D(\boldsymbol{r}_i)] + \mathbb{E}[D(\boldsymbol{r}_i-\delta\boldsymbol{u}) - D(\boldsymbol{r}_i)]\right)$$

is in $O(T^{3/4})$. Therefore, to prove that the regret of gradient sampling is in $O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right)$, it suffices to show that

$$\sum_{i=1}^{T/2\tau} 2\tau\cdot\mathbb{E}[D(\boldsymbol{r}_i) - D(\boldsymbol{r}^*)] = O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right). \tag{14}$$

As $\boldsymbol{r}^*\in\Lambda_\epsilon$ and Λ_ϵ is a convex set, we have based on Line 10 of **Algorithm 1** (following the standard analysis framework of projected gradient descent) that

$$||\boldsymbol{r}_{i+1} - \boldsymbol{r}^*||^2 = ||\Pi_{\Lambda_\epsilon}[\boldsymbol{r}_i - \eta\hat{\nabla}D(\boldsymbol{r}_i)] - \boldsymbol{r}^*||^2 \tag{15}$$

$$\leq ||[\boldsymbol{r}_i - \eta\hat{\nabla}D(\boldsymbol{r}_i)] - \boldsymbol{r}^*||^2 \tag{16}$$

$$=||\boldsymbol{r}_i - \boldsymbol{r}^*||^2 + \eta^2||\hat{\nabla}D(\boldsymbol{r}_i)||^2 - 2\eta\hat{\nabla}D(\boldsymbol{r}_i)\cdot(\boldsymbol{r}_i - \boldsymbol{r}^*) \tag{17}$$

Recall the approximate version G of D in Lemma 3, we further decompose the right-hand-side of (17) as

$$\begin{aligned}&||\boldsymbol{r}_{i+1}-\boldsymbol{r}^*||^2\\ \leq&||\boldsymbol{r}_i-\boldsymbol{r}^*||^2+\eta^2||\hat{\nabla}D(\boldsymbol{r}_i)||^2-2\eta\nabla G(\boldsymbol{r}_i)\cdot(\boldsymbol{r}_i-\boldsymbol{r}^*)\\ &+2\eta||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||\cdot||(\boldsymbol{r}_i-\boldsymbol{r}^*)||\end{aligned}\tag{18}$$

Rearranging the terms, we have

$$\begin{aligned}&2\eta\nabla G(\boldsymbol{r}_i)\cdot(\boldsymbol{r}_i-\boldsymbol{r}^*)\\ \leq&||\boldsymbol{r}_i-\boldsymbol{r}^*||^2-||\boldsymbol{r}_{i+1}-\boldsymbol{r}^*||^2+\eta^2||\nabla\hat{D}(\boldsymbol{r}_i)||^2\\ &+2\eta||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||\cdot||(\boldsymbol{r}_i-\boldsymbol{r}^*)||\end{aligned}\tag{19}$$

As $\tilde{D}$ is a convex function, we have $\nabla G(\boldsymbol{r}_i)\cdot(\boldsymbol{r}_i-\boldsymbol{r}^*)\geq G(\boldsymbol{r}_i)-G(\boldsymbol{r}^*)$. Thus, telescoping (19) and taking expectation of both sides, we obtain that

$$\begin{aligned}&2\eta\sum_{i=1}^{T/(2\tau)}\mathbb{E}[G(\boldsymbol{r}_i)-G(\boldsymbol{r}^*)]\\ \leq&\mathbb{E}[||\boldsymbol{r}_1-\boldsymbol{r}^*||^2]-\mathbb{E}[||\boldsymbol{r}_{\frac{T}{2\tau}}-\boldsymbol{r}^*||^2]+\sum_{i=1}^{\frac{T}{2\tau}}\eta^2\mathbb{E}[||\nabla\hat{D}(\boldsymbol{r}_i)||^2]\\ &+2\eta\cdot\sum_{i=1}^{T/(2\tau)}\mathbb{E}[||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||\cdot||(\boldsymbol{r}_i-\boldsymbol{r}^*)||]\end{aligned}\tag{20}$$

We now proceed to bound each terms on the right-hand-side of (20). First, for the term $\sum_{i=1}^{T/(2\tau)}\mathbb{E}[||\nabla\hat{D}(\boldsymbol{r}_i)||^2]$, note that as the routing vectors used by the gradient sampling algorithm is in Λ_ϵ over the whole time horizon, by Lemma 5 in [30], we have that with probability at least $1-O(1/T)$, $\forall t,\ \sum_{e\in\mathcal{E}}Q_e(t)=O(\ln T/\epsilon)$, and $\mathbb{E}[\sum_{e\in\mathcal{E}}Q_e(t)]=O(1/\epsilon)$ for all t. Since the regret of the policy is at most $O(T)$, we can ignore the sample paths of probability $O(1/T)$ where the statement $\forall t,\ \sum_{e\in\mathcal{E}}Q_e(t)=O(\ln T/\epsilon)$ does not hold, as those sample paths will contribute at most $O(1)$ to the regret. Therefore, we have the bound $\hat{D}(\boldsymbol{r}_i\pm\delta\boldsymbol{u})=O(\ln T/\epsilon)$ for all episode i as each $\hat{D}$ is essentially an observation of total queue length. It follows that for each episode i,

$$\begin{aligned}&\mathbb{E}[||\hat{\nabla}D(\boldsymbol{r}_i)||^2]=\mathbb{E}\left[\frac{K[\hat{D}(\boldsymbol{r}_i+\delta\boldsymbol{u})-\hat{D}(\boldsymbol{r}_i-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}\right]^2\\ \leq&O\left(\frac{\ln^2 T}{\delta^2\epsilon^2}\right)=O(\sqrt{T}\ln^2 T/\epsilon^2).\end{aligned}\tag{21}$$

Next, for the third tern on the right-hand-side of (20), note that since $\forall t,\ \sum_{e\in\mathcal{E}}Q_e(t)=O(\ln T/\epsilon)$, we have that $\tau=O(\ln^2 T/\epsilon)\geq f(\boldsymbol{Q}(t))$ for all t and sufficiently large T. Therefore, we can invoke Proposition 2 and obtain that for any episode i starting at time slot t_i,

$$\begin{aligned}&\mathbb{E}[|\hat{D}(\boldsymbol{r}_i+\delta\boldsymbol{u})-D(\boldsymbol{r}_i+\delta\boldsymbol{u})|]\\ =&\mathbb{E}[|\sum_{e\in\mathcal{E}}Q_e(t_i+\tau)-D(\boldsymbol{r}_i+\delta\boldsymbol{u})|]\\ \leq&O(\exp(-a_3\ln^2 T/\epsilon))=O(1/T).\end{aligned}\tag{22}$$

From Lemma 3 and comparing the form of the gradient estimator used in gradient sampling and the gradient of $\nabla\tilde{D}(\boldsymbol{r})=\mathbb{E}_{\boldsymbol{u}\in\mathcal{B}_K}\frac{K[D(\boldsymbol{r}+\delta\boldsymbol{u})-D(\boldsymbol{r}-\delta\boldsymbol{u})]\cdot\boldsymbol{u}}{2\delta}$, we obtain that

$$\mathbb{E}[||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||]\tag{23}$$

$$\leq\frac{\mathbb{E}[|\hat{D}(\boldsymbol{r}_i+\delta\boldsymbol{u})-D(\boldsymbol{r}_i+\delta\boldsymbol{u})|+|\hat{D}(\boldsymbol{r}_i-\delta\boldsymbol{u})-D(\boldsymbol{r}_i-\delta\boldsymbol{u})|]}{2\delta/K}.\tag{24}$$

Let d be the diameter of Λ_ϵ. It follows from combining (22) and (24) that

$$\begin{aligned}&\mathbb{E}[||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||\cdot||(\boldsymbol{r}_i-\boldsymbol{r}^*)||]\\ \leq&d\cdot\mathbb{E}[||\hat{\nabla}D(\boldsymbol{r}_i)-\nabla G(\boldsymbol{r}_i)||]\\ \leq&\frac{Kd}{2\delta}\cdot\mathbb{E}[|\hat{D}(\boldsymbol{r}_i+\delta\boldsymbol{u})-D(\boldsymbol{r}_i+\delta\boldsymbol{u})|]\\ &+\frac{Kd}{2\delta}\cdot|\hat{D}(\boldsymbol{r}_i-\delta\boldsymbol{u})-D(\boldsymbol{r}_i-\delta\boldsymbol{u})|]\end{aligned}\tag{25}$$

$$\leq O\left(\frac{1}{T\delta}\right)=O(1/T^{3/4}).\tag{26}$$

Plugging (21) and (26) into (20), we obtain that

$$\begin{aligned}&2\eta\sum_{i=1}^{T/(2\tau)}\mathbb{E}[G(\boldsymbol{r}_i)-\tilde{D}(\boldsymbol{r}^*)]\\ \leq&O(1)+\eta^2\cdot\frac{T}{2\tau}\cdot O(\sqrt{T}\ln^2 T/\epsilon)+2\eta\cdot\frac{T}{2\tau}\cdot O(1/T^{3/4}).\end{aligned}\tag{27}$$

Plugging in the value of $\eta=\epsilon/T^{3/4}$, it follows that

$$\sum_{i=1}^{T/2\tau}2\tau\cdot\mathbb{E}[G(\boldsymbol{r}_i)-G(\boldsymbol{r}^*)]=O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right).\tag{28}$$

Finally, since for all $\boldsymbol{r}$, $|G(\boldsymbol{r})-D(\boldsymbol{r})|=O(\delta)=O(1/T^{1/4})$. Thus, we have

$$\begin{aligned}&\sum_{i=1}^{T/2\tau}2\tau\mathbb{E}[D(\boldsymbol{r}_i)-D(\boldsymbol{r}^*)]\leq\sum_{i=1}^{T/2\tau}2\tau\cdot\mathbb{E}[G(\boldsymbol{r}_i)-G(\boldsymbol{r}^*)]\\ &+\sum_{i=1}^{T/2\tau}2\tau\cdot(\mathbb{E}|G(\boldsymbol{r}_i)-D(\boldsymbol{r}_i)|+\mathbb{E}|G(\boldsymbol{r}^*)-D(\boldsymbol{r}^*)|)\\ =&O\left(\frac{T^{3/4}\ln^2 T}{\epsilon}\right),\end{aligned}\tag{29}$$

which combined with (14) conclude the proof.

V. Empirical Evaluation

In this section, we conduct simulations to evaluate the empirical performance of the gradient sampling algorithm. We will test the gradient sampling algorithm and compare its performance with other baseline policies on networks with progressively more complex topology. The results on networks with simple topology are more interpretable and yield more insights on the behavior of the gradient sampling algorithm, while the results on general networks demonstrate the efficacy of the algorithm on more complex topology. In the simulations,

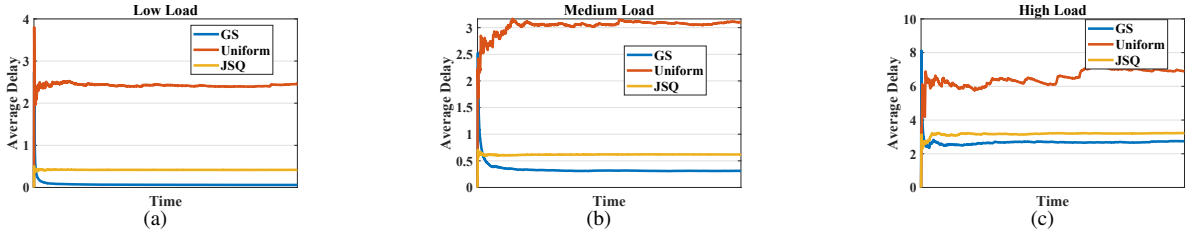


Fig. 2. Average Delay under Different Policies in Networks of Parallel Links.

the initial routing vector of the gradient sampling algorithm is set to be uniform over all paths.

A. Scenario I: Parallel Links

We first consider a network consists of three parallel links labeled 1, 2 and 3. The service rate of each link is 5 (**Figure 1**), thus the maximum supportable arrival rate of the network is equal to 15. The time horizon $T = 100000$.

1) Simulation Setup: The arrival process is deterministic and we run the simulations in the following three cases:

- *Low Load:* $a(t) = 4$ for all t.
- *Medium Load:* $a(t) = 8$ for all t.
- *High Load:* $a(t) = 12$ for all t.

Although the links have the same service rate, each link has a different service process, which we will refer to as its type:

- *Type I (deterministic):* $c_1(t) = 5$ for all t.
- *Type II (Poisson):* $c_2(t)$ is a Poisson random variable with mean 5.
- *Type III (bursty):* $c_3(t) = 40$ with probability $\frac{1}{8}$ and $c_3(t) = 0$ with probability $\frac{7}{8}$.

We run the following three policies on the network with parallel links:

- *Uniform:* route the traffic uniformly on each link, i.e., $r_1 = r_2 = r_3 = \frac{1}{3}$.
- *Join-the-Shortest-Queue (JSQ):* route the traffic to the link with the smallest queue length at each time. Note that JSQ is a dynamic policy.
- *Gradient Sampling (GS):* the gradient sampling algorithm with $\tau = 50, \delta = 0.02, \eta = 0.05, \epsilon = 0.02$.

2) Simulation Results: We plot the evolution of average packet delay with time under the three policies in the three cases of different network loads in **Figure 2**. We can see from **Figure 2** that the gradient sampling algorithm outperforms uniform and JSQ policies and the advantage of the gradient sampling algorithm decreases as the network load increases. The superiority of the gradient sampling algorithm can be attributed to its ability to learn the link types. Indeed, in the network with parallel link, one should maximally use the deterministic link and avoid the bursty link when possible. When the load is low, the optimal policy is to send all the packets on the deterministic link. However, neither the uniform policy, nor JSQ, which is a widely-used policy for routing and load-balancing, are capable of learning the link types and avoiding sending packets on the Poisson and the bursty links. Note that the observation that gradient sampling can achieve lower delay than JSQ shows the significance learning

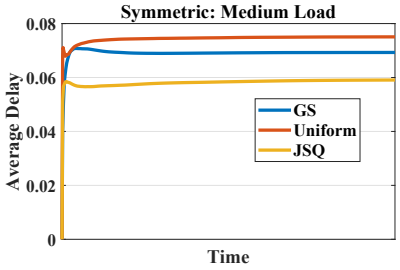


Fig. 3. Gradient Sampling Cannot Outperform the Optimal Dynamic Policy.

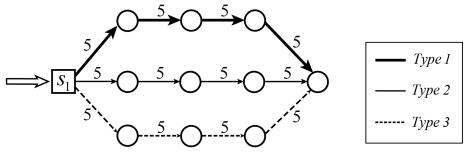


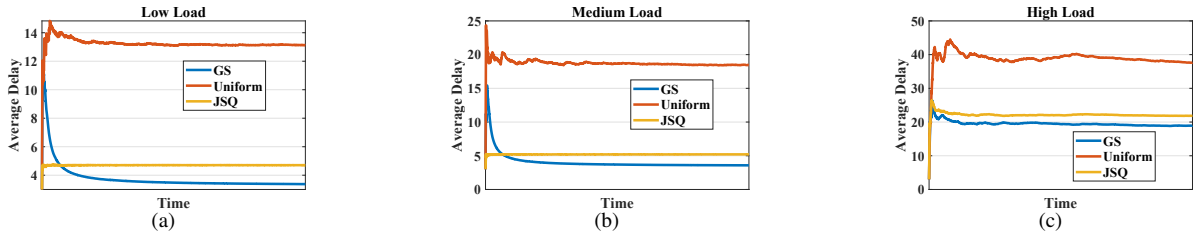
Fig. 4. Network of Parallel Paths.

the network characteristics, as even though a static policy cannot adapt to queue length variation as dynamic policies do, a static policy tailored to link types can outperform good dynamic policies.

Finally, we note that the gradient sampling algorithm aims to find the optimal static routing policy, and thus it cannot outperform the optimal dynamic policy. We run an additional simulations on a symmetric network with three links where each link is of type II under medium load, and plot the results in **Figure 3**. In this symmetric network, JSQ has been proved to be the optimal dynamic policy. From **Figure 3**, we can see that gradient sampling cannot achieve lower delay than JSQ, since the space of the static policies is smaller than the space of dynamic policies.

B. Scenario II: Parallel Paths

We next consider a network with three parallel (non-overlapping) paths, each consisting of four links of the same type. The service rate of each link is 5 (See **Figure 4**). For each path, we will refer to its type by the type of its links. Similar as in the scenario of parallel links, we run the simulations in the three cases where the loads are low ($a(t) = 4$), medium ($a(t) = 8$) and high ($a(t) = 12$). We test three policies on the network: uniform, (generalized) JSQ and gradient sampling, where the uniform and gradient sampling policies are the same as scenario I. The generalized JSQ policy routes the traffic along the path with the minimum total queue length, which is essentially a multi-hop generalization of the JSQ policy. The time horizon $T = 100000$.



(a) (b) (c)

Fig. 5. Average Delay Under Different Policies in Networks of Parallel Paths.

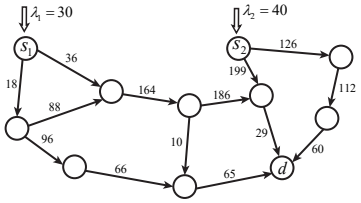


Fig. 6. The Abilene Network

We plot the evolution of average packet end-to-end delay with time under the three policies in the three cases of different loads in **Figure 5**. We can see that similar to Scenario I, gradient sampling algorithm outperforms the other two policies. Furthermore, as in the low load case the optimal static policy can be shown to be $r_1 = 1, r_2 = 0, r_3 = 0$, we validate that the gradient sampling algorithm converges to the optimal static routing policy. As the condition that the analysis in Section IV relies on has been shown to hold for networks with non-overlapping paths, our empirical results corroborates the theoretical analysis on the optimality of gradient sampling algorithm.

C. Scenario III: General Topology

Finally, we consider a network of general topology. We adopt the topology of the Abilene network of the Internet backbone [34]. The topology is shown in **Figure 6**.

1) Simulation Setup: We study a multi-commodity setting with two sources s_1, s_2 sending traffic to a common destination d. There are four paths available from s_1 to d and two paths available from s_2 to d and the paths overlap. The arrival process to s_1 is a Poisson process with rate 30 and the arrival process to s_2 is a Poisson process with rate 40. The service process of each link is Poisson with rate labeled beside the link in **Figure 6**. Note that the service rates are scaled from the original link capacities of the Abilene network [34]. The time horizon $T = 100000$.

We run three policies on the Abilene network:

- *Uniform:* both s_1 and s_2 routes the traffic uniformly along their paths to the destination.
- *Universal Max-Weight (UMW) [17]:* throughput-optimal dynamic routing policy proposed in [17].
- *BackPressure (BP):* throughput-optimal dynamic routing policy proposed in [35].
- *Gradient Sampling:* same parameters as in Scenario I.

2) Simulation Results: We plot the evolution of average end-to-end delay with time under the four policies in **Figure 7**. From **Figure 7**, we see that gradient sampling outperforms the other three policies. Although BP and UMW are dynamic policies, there is no guarantee on their delay performances and their delay performance is indeed worse than gradient sampling.

Note that in this scenario there are multiple commodities in the network and the paths overlap. Therefore, Condition 1 in Section IV may not hold and the scenario is not covered in our previous theoretical analysis. The simulation results thus show the potential of the gradient sampling algorithm in more general networks than what can be shown in theory.

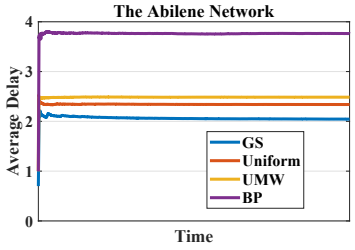


Fig. 7. Average Delay Under Different Policies in the Abilene Network.

D. Initialization for Gradient Sampling

It can be seen from our simulation results that it takes time for the gradient sampling to converge. The convergence time should depend on the distance between the initialization and the optimal routing vector. Therefore, a better initialization (than the uniform routing vector) should be able to reduce the convergence time of the gradient sampling. We validate this idea by initializing the gradient sampling with the routing vectors corresponding to the time-averages of the dynamic policies (JSQ and UMW), as those time-averages have better delay performance than the uniform routing. Such procedure indeed speeds up the convergence of gradient sampling, however, the plots are omitted due to space limitation.

VI. Conclusion

In this paper, we studied the minimum delay routing problem in stochastic queueing networks where the delay function is unknown in advance. We designed the gradient sampling algorithm that learns the values of the delay function through finite-time observations of the network queue lengths, uses the function values to construct approximate gradients of the delay function, and performs gradient descent based on the approximate gradients. The gradient sampling algorithm was shown to converge to the optimal static routing policy when the delay function is convex. It also demonstrated superior delay performance in networks of various topology in simulations.

REFERENCES

[1] M. Bramson. "Convergence to equilibria for fluid models of FIFO queueing networks.", in *Queueing Systems*, Vol. 22, No. 1, pp: 5-45, 1996.

[2] M. Neely, "Stochastic network optimization with application to communication and queueing systems.", Synthesis Lectures on Communication Networks 3, No. 1, 2010.

[3] D. Cantor and Ma. Gerla. "Optimal routing in a packet-switched computer network." in *IEEE Transactions on computers*, Vol. 23, No. 10, pp: 1062-1069, 1974.

[4] R. Gallager. "A minimum delay routing algorithm using distributed computation." in *IEEE transactions on communications*, Vol. 25, No. 1, pp: 73-85, 1977.

[5] S. Vutukury and J. J. Garcia-Luna-Aceves. "A simple approximation to minimum-delay routing." in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp: 227-238, 1999.

[6] D. Bertsekas, E. Gafni, and R. Gallager. "Second derivative algorithms for minimum delay distributed routing in networks." in *IEEE Transactions on Communications*, Vol. 32, No. 8, pp: 911-919, 1984.

[7] C. Cassandras, M. Vasmi Abidi, and D. Towsley. "Distributed routing with on-line marginal delay estimation." in *IEEE Transactions on Communications*, Vol. 38, No. 3, pp: 348-359, 1990.

[8] J. Schmitt, F. Zdarsky, and M. Fidler. "Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch..." in *IEEE INFOCOM*, pp: 1669-1677, 2008.

[9] M. Shafiee and J. Ghaderi. "A simple congestion-aware algorithm for load balancing in datacenter networks." in *IEEE/ACM Transactions on Networking*, Vol. 25, No. 6, pp: 3670-3682, 2017.

[10] R. Urgaonkar and M. J. Neely. "Optimal routing with mutual information accumulation in wireless networks." in *IEEE Journal on Selected Areas in Communications*, Vol. 30, No. 9, pp: 1730-1737, 2012.

[11] P. Djukic and S. Valaee. "Delay aware link scheduling for multi-hop TDMA wireless networks." in *IEEE/ACM Transactions on Networking*, Vol. 17, No. 3, pp: 870-883, 2008.

[12] M. Harchol-Balter. "Performance modeling and design of computer systems: queueing theory in action." Cambridge University Press, 2013.

[13] J. Van Mieghem. "Dynamic scheduling with convex delay costs: The generalized c— mu rule." in *The Annals of Applied Probability*, pp: 809-833, 1995.

[14] S. Saghafian and M. Veatch. "A c-mu rule for two-tiered parallel servers." in *IEEE Transactions on Automatic Control*, Vol. 61, No. 4, pp: 1046-1050, 2015.

[15] B. Ji, C. Joo, and N. Shroff. "Delay-based back-pressure scheduling in multihop wireless networks." in *IEEE/ACM Transactions on Networking*, Vol. 21, No. 5, pp: 1539-1552, 2012.

[16] C. Ying, E. Yeh, and R. Liu. "Enhancing the delay performance of dynamic backpressure algorithms." in *IEEE/ACM Transactions on Networking*, Vol. 24, No. 2, pp: 954-967, 2015.

[17] A. Sinha and E. Modiano. "Optimal control for generalized network-flow problems." in *IEEE/ACM Transactions on Networking*, Vol. 26, No. 1, pp: 506-519, 2017.

[18] P. Jhunjhunwala and S. T. Maguluri. "Low-complexity switch scheduling algorithms: delay optimality in heavy traffic." in *IEEE/ACM Transactions on Networking*, 2021.

[19] M. Neely and S. Supittayapornpong. "Dynamic Markov decision policies for delay constrained wireless scheduling." in *IEEE Transactions on Automatic Control*, Vol. 58, No. 8, pp: 1948-1961, 2013.

[20] P. Pinyoanuntapong, M. Lee, and P. Wang. "Delay-optimal traffic engineering through multi-agent reinforcement learning." in *IEEE INFOCOM Workshops*, pp: 435-442, 2019.

[21] Q. Liang and E. Modiano. "Optimal network control in partially-controllable networks." in *IEEE INFOCOM*, pp: 397-405, 2019.

[22] W. Tsai, J. K. Antonio, and G. M. Huang. "Complexity of gradient projection method for optimal routing in data networks." in *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, pp: 897-905, 1999.

[23] O. Shamir and T. Zhang. "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes." in *International Conference on Machine Learning*, pp: 71-79, 2013.

[24] A. Flaxman, A. T. Kalai, and H. B. McMahan. "Online convex optimization in the bandit setting: gradient descent without a gradient." arXiv preprint cs/0408007 (2004).

[25] S. Bubeck. "Convex optimization: Algorithms and complexity." in *Foundations and Trends in Machine Learning*, Vol. 8, No. 3-4, pp: 231-357, 2015.

[26] X. Fu and E. Modiano. "Learning-NUM: Network Utility Maximization with Unknown Utility Functions and Queueing Delay." in *ACM Mobihoc*, pp: 21-30, 2021.

[27] T. Chen and G. Giannakis. "Bandit convex optimization for scalable and dynamic IoT management." in *IEEE Internet of Things Journal*, Vol 6, No. 1, pp: 1276-1286, 2018.

[28] D. Down and S. P. Meyn. "Piecewise linear test functions for stability and instability of queueing networks." in *Queueing Systems*, Vol. 27, No. 3, pp: 205-226, 1997.

[29] G. Fayolle, V. Malyshev, and M. Menshikov. "Topics in the constructive theory of countable Markov chains." Cambridge university press, 1995.

[30] H. Yu, M. Neely, and X. Wei. "Online convex optimization with stochastic constraints." in *Advances in Neural Information Processing Systems*, 2017.

[31] L. Gun, A. Jean-Marie, A. Makowski, and T. Tedijanto. "Convexity results for parallel queues with bernoulli routing." 1990.

[32] M. Neely and Eytan Modiano. "Convexity in queues with general inputs." in *IEEE Transactions on Information Theory*, Vol. 51, No. 2, pp: 706-714, 2005.

[33] M. Shaked and J. Shanthikumar. "Stochastic convexity and its applications." in *Advances in Applied Probability* Vol. 20, No. 2, pp: 427-446, 1998.

[34] Network, Abilene Backbone. "Internet2." (2003).

[35] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *IEEE Conference on Decision and Control*, Vol. 4, pp: 2130-2132, 1990.