Fast Detection of Elephant Flows with Dirichlet-Categorical Inference

Aditya Gudibanda, Jordi Ros-Giralt, Richard Lethin
Reservoir Labs
632 Broadway Suite 803
New York, NY 10012
{gudibanda,giralt,lethin}@reservoir.com

***Abstract*— The problem of elephant flow detection is a longstanding research area with the goal of quickly identifying flows in a network that are large enough to affect the quality of service of smaller flows. Past work in this field has largely been either domain-specific, based on thresholds for a specific flow size metric, or required several hyperparameters, reducing their ease of adaptation to the great variety of traffic distributions present in real-world networks. In this paper, we present an approach to elephant flow detection that avoids these limitations, utilizing the rigorous framework of Bayesian inference. By observing packets sampled from the network, we use *Dirichlet-Categorical inference* to calculate a posterior distribution that explicitly captures our uncertainty about the sizes of each flow. We then use this posterior distribution to find the most likely subset of elephant flows under this probabilistic model. Our algorithm rapidly converges to the optimal sampling rate at a speed $O(1/n)$, where n is the number of packet samples received, and the only hyperparameter required is the targeted *detection likelihood*, defined as the probability of correctly inferring all the elephant flows. Compared to the state-of-the-art based on static sampling rate, we show a reduction in error rate by a factor of 20 times. The proposed method of Dirichlet-Categorical inference provides a novel, powerful framework to elephant flow detection that is both highly accurate and probabilistically meaningful.**

I. Introduction

Since its inception, the Internet has experienced a remarkable growth due to one simple but critical design paradigm: *simplicity*. This principle ensures that the core of the network is kept *scalable*, maximizing network availability and robustness. As a sign of its success, today more than 20 billion devices are connected to the global Internet delivering unprecedented levels of connectivity. Yet while this principle has been instrumental to ensure the network's scalability, it has come with a cost manifested by the network's inability to guarantee the *Quality of Service (QoS)* experienced by the applications.

From this perspective, the trade-off of scalability versus QoS defines a spectrum of possible Internet architectures. On one edge of this spectrum, scalability is maximized by using the simplicity paradigm, at the cost of providing no QoS. On the other edge, complexity is added to the core of the network to implement QoS, at the cost of severely limiting scalability. A good architecture is thus one that exploits arbitrage conditions found within this spectrum, i.e., sweet-spots that allow to significantly increase QoS (or scalability) without substantially decreasing scalability (or QoS).

The problem of *elephant flow detection* can be understood as an attempt to identify one of these sweet-spot regions. *Elephant flows* are informally defined as flows in a network that are large enough (e.g., high-rate flows or high byte count flows) to negatively affect the QoS of the smaller *mouse flows*. Because of the lack of QoS support, IP routers don't have a good way to differentiate large flows from small flows. Thus, the presence of one or just a few elephant flows can drastically deteriorate the QoS of a large number of mouse flows if they share the same router resource. One approach to resolve this issue is to implement per-flow queuing, a solution in which routers maintain a separate queue for each flow. While this ensures fine-grained control of the QoS assigned to each flow independently, its implementation is unscalable as the number of flows grows unbounded. Instead, a sweet-spot solution consists in separating the elephants and the mice into two separate queues and apply different QoS policies to them (e.g., assigning higher priority to the mouse queue). This approach can substantially help improve the network's QoS by ensuring all mouse flows are protected while still preserving its scalability properties.

A main challenge to protect QoS-sensitive mouse flows resides then around the problem of rapid detection of elephant flows so they can be properly isolated from each other. The difficulty in performing such task is once again scalability. In its general form, separating elephant and mouse flows requires measuring the size of each flow, and such task requires per-flow state, which does not scale.

To address this challenge, in this paper we take an information theoretic approach to the problem of elephant flow detection. Specifically, we tackle the question of identifying the minimum amount of information needed to detect the elephant flows with high probability. As shown in [1], it turns out that due to the property of heavy tail traffic, it is possible to identify the top largest flows without the need to inspect every single packet. This idea is similar to the concept of the *Nyquist sampling rate*, which states that a signal can be reconstructed from just a subset of samples from such signal. While [1] provides exact mathematical equations to compute the optimal sampling rate for the

correct detection of elephant flows given a target detection likelihood, these equations are shown to be impractical in that they assume an oracle view with full information of the network's past and future state. In this work, we tackle the problem of computing optimal sampling rates from a practical standpoint by using the method of *Bayesian inference* on a categorical likelihood function. Specifically, we will show that it is possible to accurately estimate the detection likelihood (and thus the optimal sampling rate) using a *Dirichlet distribution* to construct a prior on the set of $|F|$ positive unit-normalized flow sizes, where F corresponds to the set of flows in the network. Our analytical framework enables also the calculation of mathematically proven time convergence bounds, a critical aspect to understand the stability of elephant flow detection algorithms. In particular, we demonstrate our proposed algorithm rapidly converges to the optimal sampling rate at a speed $O(1/n)$, where n is the total number of packets observed, and a significant reduction in the detection error rate compared to state-of-the-art static sampling algorithms. While often under-exploited outside the field of statistics, the Dirichlet distribution provides a powerful analytical framework to address the problem of reconstructing the statistical properties of random signals from a limited amount of information [2], [3], [4], [5].

II. Related Work

Research on the problem of elephant flow detection has focused on two orthogonal problems. On one hand, there is a need to design scalable data structures to keep track of the necessary flow state. Examples of this line of work include *counter arrays* [6], [7] or sketching data structures such as *count-min sketch* [8]. Also in this area and more recently, [9] proposes a data structure based on a double hash table that is asymptotically optimal in space and time complexity. On the other hand, a second line of research has dealt with the problem of *analytical detection*, i.e., developing new mathematical models to efficiently infer the set of elephant flows. Our work focuses on this latter problem, for which we next summarize some of the previous work.

Traditional approaches to the analytical problem of elephant flow detection have focused around the identification of key flow metrics that are good quality indicators of the presence of large flows. For instance, Yi Lu et al. use both rate and byte-count thresholds to detect elephant flows [10]. Others have also used metrics such as flow rate [11] [12] [13], duration [11] [12] or burstiness [11] [12] [14]. One general issue with these metric-based approaches is that they can lead to detection errors, because they provide no mathematically formal way to identify optimal thresholds that work for all traffic distributions. Our work differs from these approaches in that by computing the optimal packet sampling rate, our algorithm can capture the top largest flows without the need to specify thresholds.

The problem of scalability by using packet sampling is also addressed in several works. For instance, Psounis et al. [15] introduce an elegant low-complexity scheduler that, based on the concept of packet sampling, detects when a flow traversing a network router or switch is likely to be an elephant flow. Those flows deemed to be elephants, are forwarded to a lower priority queue so that their presence does not deteriorate the QoS of the mouse flows. In [10], the idea of packet sampling is generalized to design an elephant trap, a data structure that can efficiently retain the elephants and evict the mice requiring low memory resources. A limitation in these algorithms is that the sampling rate is assumed to be a fixed input. However as proven in [1], if the traffic is non-stationary, a fixed sampling rate can lead to either false positives (if the sampling rate is too high) or false negatives (if the sampling rate is too low). Our approach differs in that we develop a mathematical framework to compute the optimal sampling rate and a scalable real time algorithm that dynamically adjusts such rate as the traffic statistical properties change. Our algorithm is robust to false positives and negatives as it also computes in real time the optimal threshold separating the elephants from the mouse flows, with rigorously proven correctness.

The problem of attempting to identify the optimal sampling rate to detect elephant flows has also been addressed by Zhang et al. [13]. Based on a Bayesian single sampling method, the authors propose an ingenious algorithm that identifies the sampling rate based on user-specified false positive and negative rates. This approach, however, requires multiple parameters to be specified including a threshold to classify the elephant flows (see Fig. 4 in [13]) and assumes the optimal sampling rate is static, which in general does not hold if the traffic is non-stationary. In our approach, users specify a target detection likelihood (instead of the rate of false positives and negatives), we require no additional parameters or thresholds, and our algorithm dynamically readjusts the optimal sampling rate on non-stationary traffic.

III. Mathematical Framework

A. Theory of Elephant Flow Detection under Partial Information

Our mathematical model is based on the work in [1], which provides a base theory of elephant flow detection under partial information. In the next paragraphs, we only introduce a brief summary of this framework needed to understand the model upon which the Dirichlet-Categorical inference method presented in this paper is constructed. For a full description of this theory, refer to [1]. We start by stating the definitions of *quantum error* and *detection likelihood*:

Definition 1: Quantum error (QER). Let $F = \{f_1, f_2, ..., f_{|F|}\}$ be a set of flows transmitting information over a data network and let $\boldsymbol{n}(t)$ be a vector such that its i-th element, $n_i(t)$, corresponds to the size of flow f_i at time t according to some metric (for instance, bytecounts or number of packets). $\boldsymbol{n}(t)$ is therefore a time-varying vector such that $n_i(t_f) = 0$ and $n_i(t_l) = \sigma_i$, where t_f and t_l are the times at which the first and the last bit of information are transmitted from any of the flows, and σ_i is the size of flow f_i at time t_l. Assume without loss of generality that $\sigma_i > \sigma_{i+1}$ and let $F_\epsilon = \{f_1, f_2, ..., f_\epsilon\}$ be the set with the ϵ largest flows according to their size $n_i(t_l) = \sigma_i$ at time t_l,

for some $\epsilon \leq |F|$. Finally, let $C_\epsilon(t)$ be a cache storing the top ϵ largest flows according to their size $n_i(t)$, $1 \leq i \leq \epsilon$, at time t. (Hence, by construction, $C_\epsilon(t_l) = F_\epsilon$.) We define the *quantum error (QER)* produced by the cache at time t as:

$$e_\epsilon(t) = \frac{|F_\epsilon \setminus C_\epsilon(t)|}{\epsilon} = \frac{|\{n_i(t) \text{ s.t. } \sigma_i \leq \sigma_\epsilon \text{ and } n_i(t) > n_\epsilon(t)\}|}{\epsilon} \quad (1)$$

Intuitively, the above equation corresponds to the number of top flows that at time t are incorrectly classified as small flows normalized so that the error is 1 if all top ϵ flows are misclassified. We note that while the assumed model in Definition 1 allows for different metric definitions to measure the size of a flow $n_i(t)$, for the rest of the paper we will assume a flow's size corresponds to the number of packets it has transmitted. (Note that the case of variable packet size can be addressed by assigning a sampling probability to each packet proportional to its size.)

Definition 2: Top flow detection likelihood. The top flow detection likelihood of a data network at time t is defined as the probability that the quantum error is zero, $P(e(t) = 0)$, i.e., equivalently, that the flow cache at time t captures the top ϵ flows, $P(C_\epsilon(t) = F_\epsilon)$. When the meaning is obvious, we will refer to this probability simply as the *detection likelihood*.

As proved in [1], the detection likelihood follows a multivariate hypergeometric distribution:

Lemma 1: Detection under partial information. The detection likelihood of a data network at time t follows a multivariate hypergeometric distribution as follows:

$$P(e_\epsilon(t) = 0) = P(C_\epsilon(t) = F_\epsilon) = \sum_{\forall \boldsymbol{n'} \in Z(t)} \frac{\prod_{\forall i} \binom{\sigma_i}{n'_i}}{\binom{\sum_{\forall i} \sigma_i}{\sum_{\forall i} n_i(t)}} \quad (2)$$

where Z(t) is the *zero quantum error region*, expressed as:

$$Z_\epsilon(t) = \{\boldsymbol{n'} \in \mathbb{N}^{|F|} \mid \sum_{\forall i} n'_i = \sum_{\forall i} n_i(t), \boldsymbol{n'} \leq_p \boldsymbol{\sigma}, n'_i > n'_j \ \forall i, j \text{ s.t. } i \leq \epsilon, j > \epsilon\} \quad (3)$$

and $a \leq_p b$ means b is at least as Pareto efficient as a.

Proof: See Appendix B. ∎

To illustrate the concept of detection likelihood, Figure 1 presents the graphs corresponding to Equation (2) for the case in which the traffic distribution has a single large flow of size m and n small flows of size 1. As shown:

- For the boundary case $m = 1$, the probability of finding the elephant flow is trivially zero, since the elephant flow is indistinguishable from the small flows.
- For $m > 1$ and as we increase the sampling rate p, the probability of finding the elephant flow increases.
- As the number of packets in the elephant flow m increases, a lower sampling rate is needed to achieve the same detection likelihood.

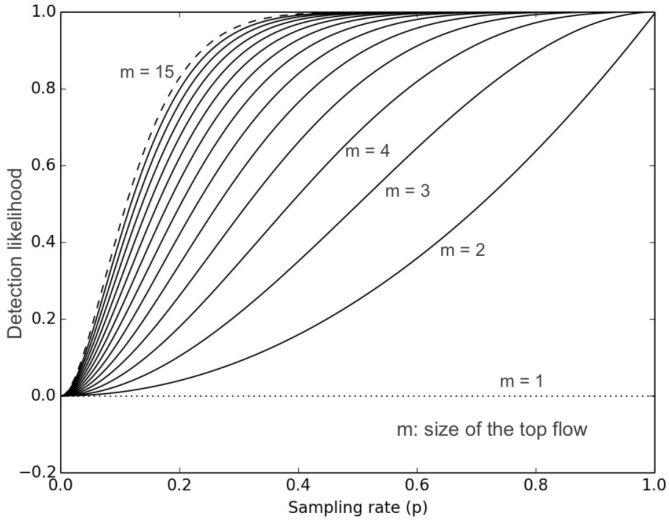


Fig. 1: Detection likelihood example.

The detection likelihood is an important concept because it indicates whether an observer can detect the elephant flows by simple inspection of the flow cache $C_\epsilon(t)$: if the detection lilkelihood is high enough, then an observer can be confident the top ϵ flows in the cache are the actual largest flows. As far as we know, the algorithm we propose in this paper is the first in the literature to tackle the problem of computing the detection likelihood, which as we will show, it enables elephant flow detection with very high precision.

Let the *sampling rate* p be defined as the fraction of data packets observed at time t, $p = \sum_{\forall i} n_i(t) / \sum_{\forall i} \sigma_i$. In [1] the authors mathematically and empirically demonstrate that, due to Internet's traffic heavy tailed properties [12], Equation (2) exposes the presence of sampling rates p_c with the following property: for $p > p_c$, the detection likelihood is close to 1; for $p < p_c$ the detection likelihood rapidly decreases to 0. Let sampling rates p_c with this property be denoted as *cutoff* sampling rates. This observation leads to the following *Reconstruction Lemma*:

Lemma 2: Flow ordering reconstruction under partial information. Let $n_i(t)$ be the size of flow f_i at time t assuming that traffic is sampled at a rate p, for $0 \leq p \leq 1$ and $1 \leq i \leq |F|$. Then the following is true:

- There exists a cutoff sampling rate p_c such that for any sampling rate $p \geq p_c$, $\sigma_i \gg \sigma_j$ implies $n_i(t) \gg n_j(t)$ with high probability.
- If the sequence $\{n_1(t), n_2(t), ..., n_{|F|}(t)\}$ is heavy tailed, then $n_i(t) \gg n_j(t)$ implies $\sigma_i \gg \sigma_j$ with high probability.
- If the sequence $\{n_1(t), n_2(t), ..., n_{|F|}(t)\}$ is not heavy tailed, then either $p < p_c$ or the traffic dataset is not heavy tailed, or both.

Proof: See [1]. ∎

The previous lemma reveals the blueprints of a class of optimal elephant flow detection algorithms. Similar to the concept of Nyquist sampling rate, the core idea is to realize that sampling traffic at a rate above p_c leads to no additional

benefits since the detection likelihood is already close to 1, while sampling traffic at rates slightly below p_c lead to significant decrease in detection likelihood. This principle implies that detection algorithms should aim at operating at a sampling rate as small as possible but not below p_c to avoid unnecessary samples while detecting elephant flows with high fidelity. In this paper we develop the first mathematically proven correct dynamic detection algorithm capable of operating at the optimal cutoff sampling rate using a Bayesian inference approach.

B. Motivation of the Bayesian Inference Approach

Let $\boldsymbol{n(t)}$ be the number of packets observed at time t, then the ratio

$$\frac{\prod_{\forall i} \binom{\sigma_i}{n_i(t)}}{\binom{\sum_{\forall i} \sigma_i}{\sum_{\forall i} n_i(t)}} \tag{4}$$

is the probability of observing $\boldsymbol{n(t)}$ at time t assuming the sizes of the flows are $\boldsymbol{\sigma} = \{\sigma_1, \cdots, \sigma_{|F|}\}$. In the framework of Bayesian inference, this is written as $P(\boldsymbol{n(t)}|\boldsymbol{\sigma})$, which is the likelihood function of our observed data $\boldsymbol{n(t)}$ given the latent flow size variables $\boldsymbol{\sigma}$.

In [1] the authors note there are two roadblocks to the computation of the detection likelihood:

- In real world network settings, for a given set of flows F, the calculation of Equation (2) is not possible at time $t < t_e$ because the true flow sizes $\boldsymbol{\sigma}$ are not known. The flow sizes $\boldsymbol{\sigma}$ encode *prior knowledge* of the true future state of the flow distribution. In other words, it assumes an unrealistic *oracle view*.
- Even if one were to know the true $\boldsymbol{\sigma}$ values, one would have to enumerate and sum over all elements of $Z(t)$, which is combinatorial in size. Thus for realistic networks the calculation of Equation (2) is computationally intractable.

In this work we solve both of these problems developing a probabilistic Bayesian framework to compute the detection likelihood by performing inference in the *opposite direction*. Instead of attempting to find $P(\boldsymbol{n(t)}|\boldsymbol{\sigma})$ by summing over all possible observations in the zero quantum error region as done in Equation (2), we seek to compute the probability distribution of the flow sizes $\boldsymbol{\sigma}$ knowing the observation $\boldsymbol{n(t)}$, $P(\boldsymbol{\sigma}|\boldsymbol{n(t)})$, and compute the detection likelihood using the information provided by the resulting posterior distributions over the flow sizes $\boldsymbol{\sigma}$. Using this approach, we achieve the two objectives: (1) a priori knowledge of the flow sizes $\boldsymbol{\sigma}$ is unnecessary and (2) the detection likelihood becomes computationally feasible to calculate. We complete the motivation of our approach by formalizing this concept into a lemma:

Lemma 3: Lower bound detection likelihood with Bayesian inference. The detection likelihood of a data network at time t in the framework of Bayesian inference of the flow sizes σ_i satisfies:

$$P(e_\epsilon(t) = 0) = P(C_\epsilon(t) = F_\epsilon) \geq \prod_{i \in F_\epsilon, j \in F \setminus F_\epsilon} P(p_i(t) > p_j(t)) \tag{5}$$

where $p_i(t)$ is the posterior marginal distribution of σ_i at time t.

Proof: Let x_i be drawn according to the distribution $p_i(t)$, for $1 \leq i \leq |F|$. For any $j \in F_\epsilon$ and $k \in F \setminus F_\epsilon$, let $X_{j,k}$ be the event that $x_j > x_k$. Given our current posterior marginal distributions for two flows, $p_i(t)$ and $p_j(t)$, the probability that we assign to the proposition that $\sigma_i > \sigma_j$ is the same as the probability that $x_i > x_j$, which is $P(p_i(t) > p_j(t))$. Thus, $P(X_{i,j}) = P(p_i(t) > p_j(t))$.

Given a set of flows F, the zero quantum error event, $C_\epsilon(t) = F_\epsilon$, corresponds to the state in which all top ϵ flows classified as elephants are larger than any of the other flows in F. In other words, $P(C_\epsilon(t) = F_\epsilon) = P\left(\bigcap X_{j,k}\right)$.

Since F_ϵ and $F \setminus F_\epsilon$ are disjoint, all the events $X_{j,k}$ are either independent or positively correlated - if one event occurs, it either makes the other events more likely (those that share a mouse flow or elephant flow) or does not affect them (those that do not share any flows in common). When conditioning on multiple events, the effect is additive - for example, given that an elephant flow x_i is greater than several other mouse flows, the probability that it will be greater than another mouse flow increases. More formally, the probability of any of the events conditioned on any subset of the others is greater than or equal to the probability of the event itself.

Thus,

$$\begin{aligned} P\left(\bigcap X_{j,k}\right) = {} & P(X_{1,\epsilon+1} | X_{1,\epsilon+2}, X_{1,\epsilon+3}, \cdots, X_{\epsilon,|F|}) \\ & \cdot P(X_{1,\epsilon+2} | X_{1,\epsilon+3}, X_{1,\epsilon+4}, \cdots, X_{\epsilon,|F|}) \\ & \cdot P(X_{1,\epsilon+3} | X_{1,\epsilon+4}, X_{1,\epsilon+5}, \cdots, X_{\epsilon,|F|}) \\ & \cdots \\ & P(X_{\epsilon,|F|-1} | X_{\epsilon,|F|}) \cdot P(X_{\epsilon,|F|}) \end{aligned}$$

where we have axiomatically expanded the probability of an intersection of events. However, because the events are positively correlated, conditioning an event on any set of events always keeps equal or increases its probability, so we get

$$\begin{aligned} P\left(\bigcap X_{j,k}\right) \geq {} & P(X_{1,\epsilon+1}) \cdot P(X_{1,\epsilon+2}) \\ & \cdot P(X_{1,\epsilon+3}) \cdots P(X_{\epsilon,|F|-1}) \cdot P(X_{\epsilon,|F|}) \end{aligned}$$

$$= \prod_{i \in F_\epsilon, j \in F \setminus F_\epsilon} P(X_{i,j}) = \prod_{i \in F_\epsilon, j \in F \setminus F_\epsilon} P(p_i(t) > p_j(t))$$

as desired. ■

Notice that in our formulation above, the only parameter is ϵ. Later in Section E we will see that the Dirichlet formulation will also allow us to compute an optimal value for ϵ, making our algorithm fully parameterless.

C. The Dirichlet-Categorical Inference Method

To facilitate the transition towards a Bayesian inference model, we assume that the flow sizes $\boldsymbol{\sigma}$ are normalized to one. In other words, we assume a normalized flow size vector $\boldsymbol{\sigma}$ such that $\sum_{\forall i} \sigma_i = 1$. This is clearly sufficient for our desired purpose of determining the relative flow ordering and, as we will see, makes our analysis tractable to techniques that avoid the combinatorial explosion of hypergeometric distributions.

We note that if we have a set of flows in a network of various unknown fixed flow sizes, then sampling a packet from the network corresponds to a probabilistic draw from a categorical distribution. Therefore, asking questions about the distribution of flow sizes has a natural reinterpretation as asking questions about the inferred posterior distribution of the corresponding categorical distribution, given the observed packets so far, $\boldsymbol{n(t)}$. The inferences we draw about the relative ordering of $\boldsymbol{n(t)}$ have a natural bijection that we will use to draw conclusions on the flow orderings of the original flow sizes $\boldsymbol{\sigma}$. From this insight, it is natural to transform the problem of detecting elephant flows into the language of performing Bayesian inference on a categorical likelihood function, where packets sampled from the network draws from this categorical distribution. Towards this objective, while often under-exploited outside the field of statistics and probability theory, the Dirichlet distribution provides a canonical choice for constructing a prior and enables several elegant properties, including conjugacy with respect to the categorical likelihood function. Because we will use it as the backbone of our method to infer the elephant flows, we formally introduce some of its key properties:

Property 1: Dirichlet's probability and marginal distributions. The Dirichlet distribution is a probability distribution over the set of categorical distributions with a fixed number of outcomes or classes, denoted by K. It is parametrized by a vector $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots \alpha_K\}$, and the support of its distribution is the set of all categorical distributions with K classes. We parametrize this family by the class probability vector $\boldsymbol{x} = \{x_1, \cdots, x_K\}$. The probability distribution of a Dirichlet distribution parametrized by a given vector $\boldsymbol{\alpha}$ is

$$Dir(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{6}$$

where $B(\boldsymbol{\alpha})$ is the multivariable Beta function, given by

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \tag{7}$$

The marginals of a Dirichlet distribution are

$$P(X_i = x_i) = B(\alpha_i, \alpha_0 - \alpha_i) \tag{8}$$

where $\alpha_0 = \sum_{i=1}^{K} \alpha_i$. This fact makes it simple to deduce in analytic form the marginals of individual variables in a Dirichlet distribution.

Property 2: Dirichlet's expectations, variances and covariances. For variables X_i, X_j which are distributed according to the marginals of the Dirichlet distribution, their expectations, variances, and covariance are given as follows:

$$\begin{aligned} \mathbb{E}[X_i] &= \frac{\alpha_i}{\alpha_0} \\ Cov(X_i, X_j) &= \frac{-\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)} \\ Var(X_i) &= \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)} \end{aligned} \tag{9}$$

We will later use the above property to demonstrate the convergence of our algorithm.

In our setting, we have a flow cache of size γ where we store the counts of the top γ flows we have observed most often. Note that in the packet sampling setting, $\gamma \ll |F|$, because we will only see a small portion of the set of the flows in the network. In the Dirichlet framework, K corresponds to the size of this flow cache, γ, and the class probability vector corresponds to the flow sizes $\boldsymbol{x} = \boldsymbol{\sigma}$. The Dirichlet distribution provides a likelihood for each possible categorical distribution which the observed samples could have been drawn from. We model our likelihood function such that the sampled packets are drawn from a categorical distribution, with each category associated with a flow, and the probability of each category in proportion to the size of the corresponding flow. Inference of the true flow size distribution is then equivalent to inference of the posterior distribution of the categorical distribution given the observed packets. More specifically, given a prior

$$p(\boldsymbol{\sigma}) = Dir(\boldsymbol{\sigma}; \boldsymbol{\alpha}) \tag{10}$$

and a set of observations

$$\boldsymbol{n} = [n_1, ..., n_\gamma] \tag{11}$$

being the counts observed from each flow, the posterior distribution is:

$$p(\boldsymbol{\sigma}|\boldsymbol{n}) = Dir(\boldsymbol{\sigma}; \boldsymbol{\alpha} + \boldsymbol{n}) \tag{12}$$

This follows from the well known fact that the Dirichlet distribution is conjugate to the categorical distribution [3], [4], [5]. Given this posterior distribution over $\boldsymbol{\sigma}$, the marginal distributions of each flow size σ_i with respect to this posterior are:

$$p_i(\sigma_i) = B(\alpha_i + n_i, \left(\sum_{i=1}^{\gamma} n_i\right) + \left(\sum_{i=1}^{\gamma} \alpha_i\right) - \alpha_i) \tag{13}$$

where we have denoted p_i as the posterior marginal distribution of σ_i, and where B is the Beta function [4], [5]. This

framework therefore provides us with a marginal posterior distribution for the size of each flow. The inferred probability that one flow is greater than another can now be obtained as follows:

$$P(\sigma_i > \sigma_j) = \int_0^1 \left(\int_x^1 p_i(y)dy \right) p_j(x)dx \qquad (14)$$

Since the distributions $p_i(\sigma_i)$ and $p_j(\sigma_j)$ are known, Equation (14) is analytically tractable. Further, we will show in later sections very fast algorithms to compute these expressions in an online, streaming manner.

To fully specify the prior, we must choose the initial $\boldsymbol{\alpha}$. We note that setting all of the α_i equal to $\frac{1}{2}$ captures the Jeffreys prior, while setting all the α_i equal to 1 corresponds to a uniform prior [5]. Either one guarantees convergence (see Lemma 5). In the experiments reported in this paper we use

$$\alpha_i = 1, \forall 1 \leq i \leq \gamma \qquad (15)$$

as this has the logical interpretation of assigning all flows a fair initial packet count of 1.

D. Online Calculation of Pairwise Probabilities

Using the Dirichlet updates from Equation (12), we can deduce the marginal posterior distributions of the flow sizes from (13). Computing the probability that one flow is greater than another is then given by (14). However, (14) is in general computationally expensive and may need to be approximated by numerical methods if implemented naively. Fortunately, there exist very fast ways to compute these stochastic inequalities in a streaming manner. In particular, we use an approach developed in [16], whose use has been primarily limited to statistics and computational biology [2].

To estimate the pairwise probabilities, from Equations (13) and (14) we need to repeatedly compute $g(a, b, c, d) = P(X > Y)$, where $X = B(a, b)$ and $Y = B(c, d)$ are beta-distributed random variables. Using [16], we are provided with the following recurrence relations:

$$\begin{aligned} g(a+1, b, c, d+1) &= g(a, b, c, d) + g_{1,0}(a, b, c, d) \\ g(a, b+1, c+1, d) &= g(a, b, c, d) + g_{0,1}(a, b, c, d) \\ g(a, b+1, c, d+1) &= g(a, b, c, d) + g_{0,0}(a, b, c, d) \end{aligned} \qquad (16)$$

where

$$\begin{aligned} g_{1,0}(a, b, c, d) &= \tfrac{h(a,b,c,d)}{a} + \tfrac{h(a+1,b,c,d)}{d} \\ g_{0,1}(a, b, c, d) &= -\tfrac{h(a,b,c,d)}{b} - \tfrac{h(a,b+1,c,d)}{c} \\ g_{0,0}(a, b, c, d) &= -\tfrac{h(a,b,c,d)}{b} + \tfrac{h(a,b+1,c,d)}{d} \end{aligned} \qquad (17)$$

and where

$$h(a, b, c, d) = \frac{B(a + c, b + d)}{B(a, b)B(c, d)} \qquad (18)$$

In our model, these equations can be interpreted as follows. We initialize all the pairwise probabilities to be $\frac{1}{2}$, and assuming a flow cache of size γ, all the initial Beta distributions are set to $B(1, \gamma-1)$. In other words, we assume we start out having observed one packet from each flow, as specified in Equation (15). As packets are observed, the parameters of the Beta distributions corresponding to these marginals are increased by one in an online manner. Assume a flow f_i is currently distributed as $B(a, b)$. Then upon receiving a new sampled packet, if it belongs to flow f_i, its new posterior is set to $B(a + 1, b)$, otherwise, its new posterior is set to $B(a, b + 1)$. We then use the recurrences in Equation (16) to compute the new resulting pairwise probabilities. These recurrences allow us to calculate the changes in the pairwise probabilities in constant time, which is a key to help scale our algorithm to real-world networks with low latency.

E. Parameterless Detection of Elephant Flows

To the best of our knowledge, all algorithms found in the literature require using a threshold to delimit the frontier that separates the elephant flows from the mouse flows (e.g., [10], [13], [15]). Our approach based on the calculation of the detection likelihood under uncertainty, however, provides a formulation to separate both types of flows in a natural manner. Recall from Definitions 1 and 2 that ϵ corresponds to the true number of elephant flows an oracle would use to compute the quantum error and the detection likelihood. While as non-oracles we cannot know ϵ, our Bayesian inference framework allows us to make an optimal guess of its value based on our partial information of the state of the network as follows. Let $\hat{\epsilon}$ be an estimator of the true value of ϵ, then we face two cases:

- If $\hat{\epsilon} < \epsilon$, then it must be that at least one elephant flow f_i is classified as mouse flow. This necessarily implies that the detection likelihood will not be maximal as it is easy to incur quantum error between any elephant flow and flow f_i.
- If $\hat{\epsilon} > \epsilon$, then it must be that at least one mouse flow f_i is classified as elephant flow. This necessarily implies that the detection likelihood will also not be maximal as it is easy to incur quantum error between any mouse flow and flow f_i.

By reduction, we must conclude that a maximal detection likelihood value is attained when $\hat{\epsilon} = \epsilon$ and thus that one can infer the oracle's value of the correct ϵ parameter using the following equation:

$$\epsilon \approx \hat{\epsilon} = \max_{arg(x)} P(e_x(t) = 0) \qquad (19)$$

That is, under the presence of uncertainty and partial information, the optimal set of elephant flows is one that maximizes the detection likelihood equation.

In practice, we use a slight modification of Equation (19) which incorporates additional information provided by a threshold on the detection likelihood, $target_{dl}$, as shown in Equation (20).

$$\epsilon \approx \hat{\epsilon} = \max\{x \mid P(e_x(t) = 0) \geq target_{dl}\} \qquad (20)$$

Using the $target_{dl}$ provides the dual benefits of (1) refraining from reporting any elephant flows if all detection likelihoods are below $target_{dl}$ and (2) making sure to classify all elephant flows that are within the required detection likelihood. These benefits reduce the problems of false positives and false negatives, respectively. We note that setting $target_{dl} = 0$ makes Equations (19) and (20) equivalent.

Equation (20) provides us with a natural way to separate elephant flows in a network based on the detection likelihood, and we will be using it in our algorithm as an optimal estimator of ϵ.

F. Dirichlet Detection Algorithm

We present the full pseudocode for the Dirichlet detection algorithm in Algorithms 1, 2, and 3. The core algorithm embedding the mathematical framework presented in this paper is Algorithm 1, process_sample(). As mentioned, we initialize all pairwise probabilities to 0.5 because our prior assumes that all marginals are initially $B(1, \gamma - 1)$. In lines 12 to 17, every pair of flows have their pairwise probability updated based on Equation (16). The vectors a and b are count vectors holding the number of times each flow was seen and not seen, respectively, and these are updated based on the sample just seen in lines 18 to 21. Once the pairwise probability a and b are all updated, the function calculate_detection_likelihoods() is called, shown in Algorithm 3. In this function, the detection likelihoods are calculated for all $\epsilon, 1 \leq \epsilon \leq \gamma$, minimizing redundant computations. The rationale behind these calculations is provided by Lemma 3, which establishes that the products calculated by Algorithm 3 are lower bounds on the detection likelihood.

In the function choose_ϵ_and_adjust_sampling_rate() (Algorithm 2), lines 3 to 5 implement the estimator for ϵ using equation (20). To ensure our choice is correct within the required accuracy target, we only select a valid value for ϵ if the detection likelihood is above the target detection likelihood $target_{dl}$. If such an ϵ exists, we decrease the sampling rate, as this means we have sufficient information to make a decision about the identities of the elephant flows, and we can try reducing the sampling rate to test whether we can still have enough information at a lower rate. If no such ϵ exists, then we currently are not receiving enough information at our sampling rate, so we increase the sampling rate to increase the information we receive in future iterations. Importantly, if there is no ϵ with a detection likelihood above the target, then we do not report the number of elephant flows, because we have not reached our desired level of confidence to make a decision. This is a critical feature of our algorithm which differentiates it from other known algorithms, and significantly reduces the quantum error rate of our algorithm, minimizing false negatives and false positives, as we will later show through real network benchmarks.

Algorithm 1: process_sample()

```
1  Initialize p[γ][γ];
2  Initialize a[γ];
3  Initialize b[γ];
4  Initialize d[γ − 1];
5  for 1 ≤ i ≤ γ do
6      for 1 ≤ j ≤ γ do
7          p[i][j] = 0.5;
8  begin
9      for sample s_i do
10         for 1 ≤ i ≤ γ do
11             for 1 ≤ j ≤ γ do
12                 if s_i is from flow i and not flow j then
13                     p[i][j] =
                         p[i][j] + g_{1,0}(a[i], b[i], a[j], b[j]);
14                 if s_i is from flow j and not flow i then
15                     p[i][j] =
                         p[i][j] + g_{0,1}(a[i], b[i], a[j], b[j]);
16                 if s_i is neither from flow i nor flow j
                     then
17                     p[i][j] =
                         p[i][j] + g_{0,0}(a[i], b[i], a[j], b[j]);
18             if s_i is from flow i then
19                 a[i] = a[i] + 1;
20             else
21                 b[i] = b[i] + 1;
22         d = calculate_detection_likelihoods();
23         choose_ϵ_and_adjust_sampling_rate();
```

Algorithm 2: choose_ϵ_and_adjust_sampling_rate()

```
1  target_dl: target detection likelihood;
2  begin
3      best_ϵ = −1;
4      for 1 ≤ ϵ ≤ γ − 1 do
5          if d[ϵ] > target_dl then
6              best_ϵ = ϵ;
7      if best_ϵ = −1 then
8          increase_sampling_rate();
9      else
10         decrease_sampling_rate();
```

G. Computational Complexity and Time Convergence

In this section we formally analyze both the computational complexity and the time convergence of the proposed algorithm.

Lemma 4: Computational Complexity. The total cost of the Dirichlet detection algorithm is $O(\gamma^2)$

Proof: The evaluation of the function h from Equation (18) takes constant time, and each pairwise probability update takes two calls to h, so they each take constant time as well. There are $O(\gamma^2)$ pairs of flows to compare. Next, for the detection likelihood calculation, there are $O(\gamma)$ detection likelihoods that need to be calculated and each takes $O(\gamma)$ time. Passing through the detection likelihoods in Algorithm

2 to choose the best ϵ takes $O(\gamma)$ time. Thus in total, our algorithm is $O(\gamma^2)$. ■

Note that since we are sampling packets at a small rate, the number of unique flows γ we see is significantly smaller than the total number of flows in the network $|F|$, $\gamma \ll |F|$. This ensures the scalability of the detection algorithm. In addition, later in Section IV we describe additional speedup optimizations.

Lemma 5: Order of Time Convergence. As the number of the packets observed n increases, under the assumption of expected asymptotic behavior, the probability of our algorithm misclassifying any flow goes to 0 at a rate of $O\left(\frac{1}{n}\right)$.

Proof: Suppose that after the observation of a set of packets, the posterior Dirichlet distribution is $Dir(\alpha_1, \cdots, \alpha_\gamma)$, and let $\alpha_0 = \sum_{i=1}^{\gamma} \alpha_i$. Since our algorithm begins with a prior of $Dir(1, \cdots, 1)$, α_0 begins at γ and therefore at any point after seeing n samples, $\alpha_0 = n+\gamma$.

Then from Equation (9),

$$\begin{aligned} Var(\sigma_i - \sigma_j) &= Var(\sigma_i) + Var(-\sigma_j) + 2Cov(\sigma_i, -\sigma_j) \\ &= Var(\sigma_i) + Var(\sigma_j) - 2Cov(\sigma_i, \sigma_j) \\ &= \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0+1)} + \frac{\alpha_j(\alpha_0 - \alpha_j)}{\alpha_0^2(\alpha_0+1)} + \frac{2\alpha_i\alpha_j}{\alpha_0^2(\alpha_0+1)} \end{aligned} \quad (21)$$

Let $r_i = \mathbb{E}(\sigma_i) = \frac{\alpha_i}{\alpha_0}$ and r_j be defined likewise. Under the assumption of expected asymptotic behavior, we substitute $r_i = \frac{\alpha_i}{\alpha_0}$ and $r_j = \frac{\alpha_j}{\alpha_0}$ in the formula for $Var(\sigma_i - \sigma_j)$ to obtain

$$\begin{aligned} Var(\sigma_i - \sigma_j) &= \frac{r_i(1-r_i) + r_j(1-r_j) + 2r_ir_j}{\alpha_0 + 1} \\ &= \frac{S_{ij}}{\alpha_0 + 1} \end{aligned} \quad (22)$$

where we have let

$$S_{ij} = r_i(1-r_i) + r_j(1-r_j) + 2r_ir_j \quad (23)$$

The probability that we misclassify the ordering of flows i and j is the probability that our algorithm assigns to $P(\sigma_j > \sigma_i) = P(\sigma_i - \sigma_j < 0)$. By Chebyshev's inequality [17],

$$P(\sigma_i - \sigma_j < 0) \quad (24)$$

$$= P((\sigma_i - \sigma_j) - (r_i - r_j) < -(r_i - r_j)) \quad (25)$$

$$< \frac{Var(\sigma_i - \sigma_j)}{Var(\sigma_i - \sigma_j) + (r_i - r_j)^2} \quad (26)$$

$$= \frac{S_{ij}/(\alpha_0+1)}{S_{ij}/(\alpha_0+1) + (r_i - r_j)^2} \quad (27)$$

$$= \frac{S_{ij}}{S_{ij} + (r_i - r_j)^2(\alpha_0+1)} \quad (28)$$

$$< \frac{S_{ij}}{(r_i - r_j)^2(\alpha_0+1)} = \frac{S_{ij}/(r_i - r_j)^2}{\alpha_0+1} = \frac{T_{ij}}{\alpha_0+1} \quad (29)$$

where we have let

$$T_{ij} = \frac{S_{ij}}{(r_i - r_j)^2} \quad (30)$$

The *total* detection likelihood is the probability that we don't misclassify *any* of the pairwise comparisons between elephant flows and mouse flows. By the union bound [17], this is bounded by the sum of the probabilities of these pairwise miscomparisons. Thus, the probability of any misclassification (equivalently, the complement of the detection likelihood), is bounded by

$$\sum_{i \in E} \sum_{j \in M} \frac{T_{ij}}{\alpha_0 + 1} \quad (31)$$

Recall that $\alpha_0 = n + \gamma$. Thus, as the number of packets observed n increases, the probability of failure goes to 0 at a rate of $O\left(\frac{1}{n}\right)$.

■

Algorithm 3: calculate_detection_likelihoods()

```
begin
    Initialize d[γ − 1];
    for 1 ≤ i < γ do
        if i = 1 then
            d[i] = 1
            for 2 ≤ j ≤ γ do
                d[1] = d[1] · p[1][j];
        else
            d[i] = d[i − 1];
            for 1 ≤ j ≤ i − 1 do
                d[i] = d[i] / p[j][i];
            for i + 1 ≤ j ≤ γ do
                d[i] = d[i] · p[i][j]
```

IV. Methods and Implementation

The implementation of the algorithm raised a number of issues and potential optimizations that needed to be addressed to scale to real-time traffic with low latency. In this section we discuss some of these implementation details.

A. Unique Count Optimization

Each pairwise probability update between flows f_i and f_j depends only upon the Dirichlet parameters for each flow, α_i and α_j. Thus, once a flow f_i's pairwise probabilities with respect to all other flows have been updated, any other flow f_j such that $\alpha_j = \alpha_i$ will have the same pairwise probability updates. Thus, to update the pairwise probabilities for f_j we simply copy the updates from f_i. This reduces the complexity of the pairwise probability updates from $O(|\gamma|^2)$ to $O(|\{\boldsymbol{\alpha}\}|)^2$, where $\{\boldsymbol{\alpha}\}$ is the set of coefficients of $\boldsymbol{\alpha}$, and $|\{\boldsymbol{\alpha}\}|$ is the number of unique elements of $\boldsymbol{\alpha}$. Because real-world traffic distributions tend to follow a power law distribution, a majority of flow counts will be concentrated in a few small numbers < 10, and this optimization leads to a significant increase in process sampling speed.

B. Ghost Flow Handshake

In real networks, flows come into and out of existence dynamically. It is thus imperative that we can incorporate both events into our algorithm. To account for flows ending, we implemented a timekeeping mechanism which detects when a flow times out. In this event, this flow is deleted from the a and b vectors, and the corresponding row and column in p are deleted as well. The problem of a new flow coming into existence poses a unique challenge. We note that a naive implementation of adding the flow to a, b and p, and calculating all the pairwise probabilities when adding to p would lead to a large delay because the method for computing pairwise probabilities is a recursion, and hence scales with the total number of packets observed. To resolve this problem, we store an extra "ghost" flow in our a, b and p data structures, and we keep the observed count of this flow to be 1. We then update this ghost flow's pairwise probabilities in the p matrix as normal. When a new flow is observed, we make a copy of this ghost flow, and then assign the new flow to the original ghost flow. Thus, the new flow will incorporate all the up-to-date computed probabilities of the ghost flow, and the copy of the ghost flow will be available to incorporate additional flows. This technique amortizes the cost of adding new flows into the streaming calculation and ensures that our algorithm has low variance in the time taken to compute the pairwise probabilities of a sampled packet, regardless of whether it is from a new, unobserved flow or not.

C. Static Sampling Implementations

As described in the analytical sections above, our algorithmic framework based on Dirichlet-Categorical inference provides two new features with respect to the state-of-the-art: (1) the ability to compute an optimal sampling rate that is as small as possible while ensuring all the elephant flows are captured in the flow cache and (2) the ability to estimate the detection likelihood and thus know the level of accuracy of the current estimation. In many network measurement scenarios, the sampling rate is fixed. For instance, the sFlow protocol [18] normally operates using a static sampling rate defined by the network operator. If the sampling rate is fixed, then there is no need to attempt to compute a dynamic optimal sampling rate, but our algorithm can still be very useful in these scenarios as it provides a mechanism to eliminate quantum error by using the detection likelihood.

Indeed, in the problem of elephant flow detection, uncertainty comes from two sources: (1) the inability to capture all traffic at line rate, therefore needing to sample traffic which leads to information loss, and (2) the inability to know the future performance of each flow. Hence there are two ways by which uncertainty can be reduced: (1) by increasing the sampling rate to inspect more packets or (2) by delaying the detection decision until more packets are collected. In the cases where the sampling rate is static, there is no need to identify the optimal sampling rate but there is still a need to identify the optimal time the algorithm needs to wait until an accurate detection decision can be made. It is in this regard that the computation of the detection likelihood provides a key piece of information to reduce noise in the detection process. In summary, if the sampling rate is static, our algorithm still provides value to the network operator as it helps determine at which point in time the detection likelihood is high enough that a classification decision can be made with a quantifiable probability of error given by the complement of the detection likelihood $1 - P(e(t) = 0)$.

Gaussian	Laplace	Sech-square	Cauchy	Linear
$\tau_i e^{-\frac{1}{2}i^2}$	$\tau_i e^{-i}$	$\frac{\tau_i e^{-i}}{(1+e^{-i})^2}$	$\frac{\tau_i}{1+i^2}$	$\tau_i(\gamma - i)$

TABLE I: Analytic forms of synthetic traffic distributions. τ_i is chosen such that $\sum_{i=1}^{\gamma} \tau_i = 300$, and the distributions are shown in order of increasing entropy.

V. Results

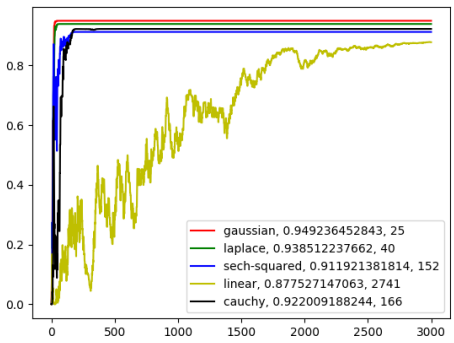
A. Synthetic Data

In our first set of experiments our objective is to empirically validate how closely the Dirichlet inference framework tracks the true oracle equations presented in [1]. Towards this goal, we measured the detection likelihood computed by our algorithm on categorical draws from the same well-known analytic distributions used in [1]: Gaussian, Laplace, Cauchy, Sech-Squared, and Linear. The flow size equations for each distribution are presented in Table I. In our simulations, flows generate a number of packets according to each distribution, rounding the number to integers by taking the ceiling.

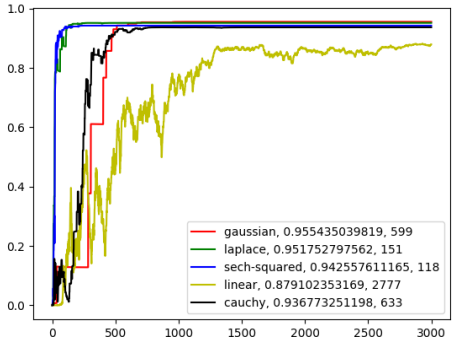
We implemented and ran Algorithms 1, 2, and 3 for the number of flows $\gamma = 10, 40, 100, 200$, and the detection likelihood was calculated for $\epsilon = 2, 3, 4, 5$. The results of the runs for $\gamma = 10$ and $\gamma = 100$ are respectively shown in Figures 2 and 3 for each of the 5 traffic distributions. Further results for $\gamma = 40$ and $\gamma = 200$ are shown in the Appendix in Figures A1 and A2.

We first test the results against Lemma 2. Since there is a linear correspondence between the sampling rate p in Lemma 2 and the number of samples observed n shown in Figures 2 and 3, we will use them interchangeably. The first statement in Lemma 2 tells us that for a heavy tailed distribution, there must be a number of samples n_c, analogous to p_c, such that the detection likelihood is high at n_c onwards, and is smaller before n_c. It is clear from Figures 2 and 3 that, for the heavy tailed traffic distributions, there is a cutoff value n_c, analogous to p_c in Lemma 2, at which the detection likelihood rapidly approaches its maximum, and additional samples do not significantly increase the detection likelihood further, just as Lemma 2 predicts.

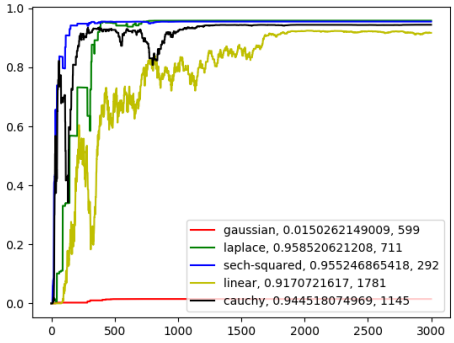
The second bullet point in Lemma 2 tells us that if our observed flow counts are heavy tailed, then the true flow distribution likely is as well. We quantify the extent to which a distribution is heavy tailed by measuring its entropy. In Figure 4, we plot the entropy of the posterior Dirichlet distribution for each input traffic distribution, for $\gamma = 10$ and $\gamma = 100$. We see that the entropies of the Dirichlet posteriors are in the same order as the entropies of the respective input



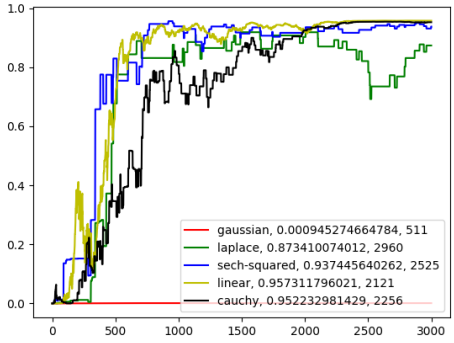
(a) $\gamma = 10$ and $\epsilon = 2$.



(b) $\gamma = 10$ and $\epsilon = 3$.



(c) $\gamma = 10$ and $\epsilon = 4$.



(d) $\gamma = 10$ and $\epsilon = 5$.

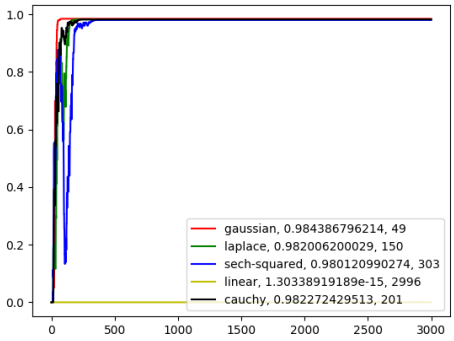
Fig. 2: Detection likelihood over the number of samples with $\gamma = 10$

distributions, which is given in Table I. Thus, for an input distribution which has lower entropy (more heavy tailed), our posterior distribution has lower entropy, meaning that our algorithm is more certain. Figures 2 and 3 also show that the time to convergence follows this same order as well, with Gaussian being the fastest to convergence, followed by Laplace, then Sech-Square, then Cauchy and ending with Linear. This result based on the Dirichlet inference model is precisely in agreement with the oracle equations plotted in Figure 3 of [1].

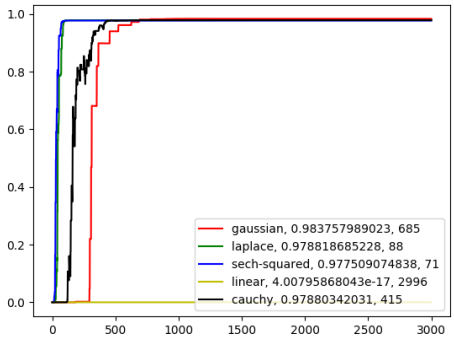
Next, the third bullet point would lead us to expect that if our observed samples are not heavy tailed, then we should be less certain about deciding upon a certain flow ordering, and instead wait and observe more samples. This behavior can indeed be seen in the Linear detection likelihood in Figure 2(a). Although we were able to deduce that σ_1 is the largest flow for the Gaussian distribution by iteration 25, it took roughly 3000 iterations to conclude the same for the Linear distribution. Since the Linear distribution is not a heavy tailed distribution, this is precisely the expected behavior.

The above analysis provides empirical evidence that the Dirichlet inference framework is capable of qualitatively following the oracle equations and the Reconstruction Lemma under the presence of uncertainty and partial information.

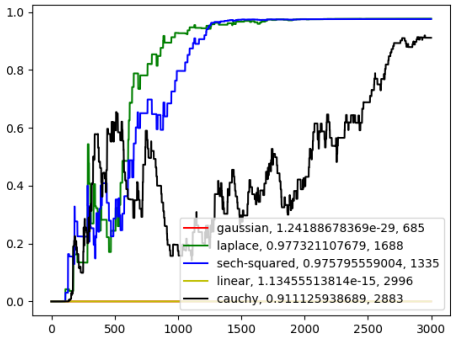
Finally, the experiments help also to empirically validate the accuracy of the estimator $\hat{\epsilon}$ according to equation (20). Consider for example the Gaussian distribution, for which the equation in Table I corresponds to the flow size distribution $[242, 54, 5, 1, 1, 1, 1, 1, 1, 1, ...]$. (Remember that in our simulations we round the flow sizes to integers by taking the ceiling.) As shown in Figures 2(a) and 3(a), our algorithm quickly identifies the first two elephant flows by iterations 25 and 49 for the cases of $\gamma = 10$ and $\gamma = 100$, respectively. With the given limited information received up until that point, the Dirichlet method tells us also that the top 2 elephant flows captured in the cache ($\hat{\epsilon} = 2$) correspond to the true top two largest flows with a probability of 94.9% for $\gamma = 10$ and 98.4% for $\gamma = 100$. If we wait to collect a bit more information, then from Figures 2(b) and 3(b), we detect a third elephant flow with a detection likelihood of 95.5% and 98.3% at iterations 599 and 685, respectively. However, notice that the Gaussian curve for the cases $\epsilon = 4$ and $\epsilon = 5$ stay flat near zero. Thus the algorithm is able to correctly infer that there exists no more than 3 elephant flows (since the probability of that being the case drops to zero as soon as we switch ϵ from 3 to 4). Using equation (20), our estimator $\hat{\epsilon}$ is able to identify the right number of elephant flows by choosing at every point in time the value of ϵ that maximizes the detection likelihood.



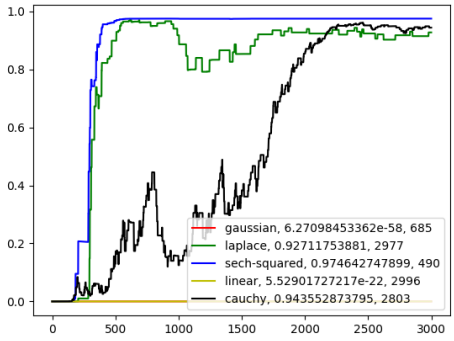
(a) $\gamma = 100$ and $\epsilon = 2$.



(b) $\gamma = 100$ and $\epsilon = 3$.



(c) $\gamma = 100$ and $\epsilon = 4$.



(d) $\gamma = 100$ and $\epsilon = 5$.

Fig. 3: Detection likelihood over the number of samples with $\gamma = 100$

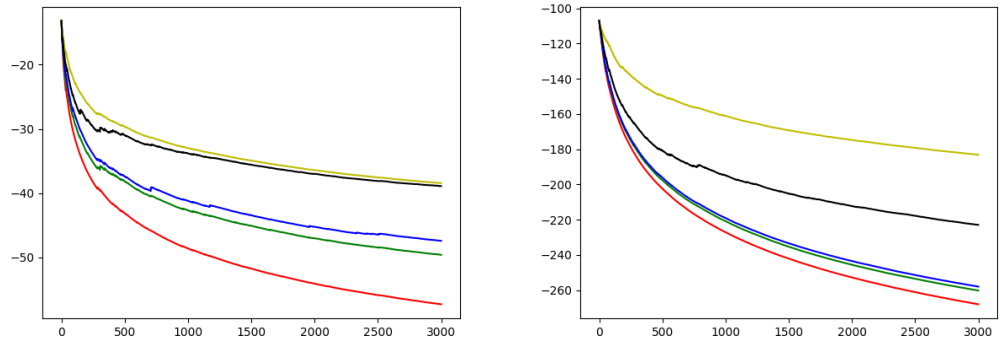
B. Network Data

In our second set of experiments the main objective is to test our algorithm in a real SDN networking environment and against some of the considered state-of-the art detection algorithms. To this end, we built an SDN test lab using Open vSwitch (OVS) for network engineering and Linux KVM for virtualization of hosts of our simulated network. The server hosting the nodes of the network had 2 Xeon E5-2670 CPUs for a total of 32 cores and 64GB of RAM. A base KVM hypervisor was used to virtualize the complete SDN test lab, and inside the base KVM, we use (1) OVS to build arbitrary configurations of SDN-capable network topologies and (2) another layer of (nested) KVMs to create arbitrary numbers of VM hosts used as sources and sinks in the SDN network. Our simulated network had two nodes and one virtual switch between them, and packets were sampled from this switch while traveling from the source node to the sink node using sFlow, the well-known standard protocol used to monitor networks using packet sampling [18]. We implemented a controller that talks with the OVS switch to control the sFlow sampling rate in real time and connected the controller to our implementation of the Dirichlet detection algorithm. Our simulated traffic alternated every 30s between a traffic distribution with five elephant flows, and another disjoint traffic distribution with ten elephant flows, to determine how our algorithm adjusts to changes in the network's traffic distribution. The two distributions did not share any flows in common.

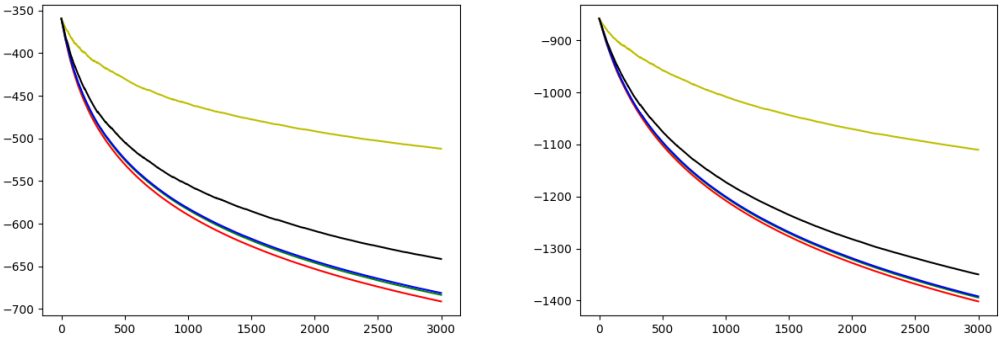
Our main objective is to demonstrate our claim about the importance of both using dynamic sampling rate and inferring the detection likelihood in order to detect elephant flows with accurate precision in dynamic real world networks. To the best of our knowledge, none of the existing well-known methods (such as the Elephant Trap [10], SIFT [15] or BSS [13], among others) are capable of computing in real time the optimal sampling rate nor inferring the detection likelihood. It is important to notice that while in the next benchmarks we will measure the performance benefits of using the Dirichlet detection algorithm, our contribution can be seen as non-competing because the algorithm we propose can be used as a standalone probabilistic building block to enhance any of the existing algorithms in the literature.

Thus as a benchmark to compare our algorithm against, we first test the performance of a static sampling approach as is done in today's sampling rate-based detection algorithms. In particular, for this base benchmark we use an arbitrary static sampling rate of 0.01, as is done in [15]. Notice however that the following results are applicable to other static sampling rate values.

The results for static sampling rate are shown in Figure 5. The algorithm fluctuates rapidly, hovering around the ground



(a) $\gamma = 10$ (b) $\gamma = 40$



(c) $\gamma = 100$ (d) $\gamma = 200$

Fig. 4: Entropy curves over the number of samples.

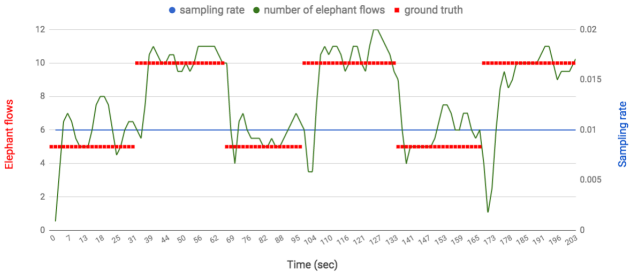


Fig. 5: The number of elephant flows detected by the static sampling algorithm over the number of samples.

truth number of elephant flows. Since the algorithm is not situated within a rigorous probabilistic framework, it cannot infer the certainty with which it reports its classifications. This leads to a non-zero quantum error rate, including both false negatives and false positives, with the algorithm reporting these errors without regard to any inherent uncertainty. Note that all static sampling algorithms which are not placed within a probabilistic framework suffer from the same problems.

Naturally, a problem with the static sampling rate framework is that the particular choice of the sampling rate is critical. More problematic also is that this choice implicitly encodes an a priori decision regarding the definition of an elephant flow. Assumptions on the size of an elephant flow, such as being more than 10% of the link capacity, are used for probabilistic justifications in [15] to bound the probability of error. However, real world network traffic varies widely and the optimal threshold for elephant flows does so as well. This has important ramifications for classification of real world network traffic, because the definition of elephant flow changes our optimization objective, which then affects the QoS. We reason that this is the wrong direction—the QoS requirements should inform our definition of elephant flows, and hence we should seek algorithms which make fewer assumptions on the definition of elephant flows, and are capable of handling a wider variety of traffic.

In Figure 6, we present the result of our algorithm in the same setting. The algorithm reports the correct ground truth number of elephant flows with high detection likelihood for all time points except two. The sampling rate, shown in blue, fluctuates from 0.05 to 0.08, and increases or decreases based on the detection likelihood. The algorithm is able to maintain a detection likelihood close to one for a large portion of the time, while it successfully stays at the minimum necessary rate to receive enough information to properly classify the elephant flows. Most importantly, in the time periods where the plot of ground truth (shown in red) does not overlap with the plot of our algorithm's elephant flow classification

Fig. 6: The number of elephant flows detected by our Dirichlet Algorithm over the number of samples.

(shown in green) our algorithm refrained from reporting the number of elephant flows, because the detection likelihood was below the target. For example, at the beginning of each shift in the traffic distribution, the detection likelihood dips and the algorithm stops reporting any elephant flow classification, just as desired. This leads to a reduction in quantum error rate, and summarizes one important contribution of the detection likelihood based method achieved by the probabilistic framework of Bayesian inference: its capability to understand its own limitations, refraining from making detection decisions when they are likely to be erroneous.

To quantify the degree of accuracy achieved by both methods, we calculate the average quantum error using Equation (1), averaged over the entire test time. The average quantum error for the static sampling benchmark and our algorithm are 0.19844 and 0.00893, respectively, and thus for the performed test our algorithm achieves a reduction in average quantum error by roughly a factor of 22.

VI. Conclusions

In this work, we have introduced a new mathematical framework for the detection of elephant flows and presented an algorithm within this framework using the theory of Bayesian statistics. This algorithm is fully parameterless, uses dynamic sampling, has mathematically proven fast convergence and achieves high classification accuracy with low overhead. We provide proof of convergence of this algorithm in $O(1/n)$ time and network benchmarks showing significant reduction in error detection rate by a factor of 20 times compared to existing algorithms based on a static sampling rate.

This paper reasons about the importance of developing analytical frameworks to compute the detection likelihood. Such a parameter turns out to be key in helping stabilize elephant flow detection algorithms in highly dynamic environments as those found in real networks. On one hand, it allows us to know the degree of certainty that the detection is correct; on the other, it allows us to know when we have not collected enough information to make a sound detection decision. Further, while the presence of optimal cut-off sampling rates has been recognized in prior work, the proposed algorithm is capable of exploiting this property in a practical manner. Such an approach leads to substantial packet processing savings while maintaining high accuracy.

We are currently working to integrate the proposed algorithm as part of an SDN network architecture for large-scale data centers. Future work will include leveraging this effort towards performing real time traffic engineering in a real data center or wide-area SDN.

References

[1] J. Ros-Giralt, A. Commike, S. Maji, and M. Veeraraghavan, "High Speed Elephant Flow Detection Under Partial Information," *International Symposium on Networks, Computers and Communications (ISNCC)*, 2018.

[2] E. Raineri, M. Dabad, and S. Heath, "A note on exact differences between beta distributions in genomic (methylation) studies," *PLoS ONE*, vol. 9, no. 5, 2014.

[3] G. Casella and R. L. Berger, *Statistical Inference*, 2002.

[4] B. a. Frigyik, A. Kapila, and M. R. Gupta, "Introduction to the Dirichlet Distribution and Related Processes," *Electrical Engineering*, vol. 27, no. 206, pp. 46–48,50–52, 2010.

[5] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, 2004.

[6] G. Einziger, B. Fellman, and Y. Kassner, "Independent counter estimation buckets," in *Proceedings - IEEE INFOCOM*, vol. 26, 2015, pp. 2560–2568.

[7] Y. Li, H. Wu, T. Pan, H. Dai, J. Lu, and B. Liu, "CASE: Cache-assisted stretchable estimator for high speed per-flow measurement," in *Proceedings - IEEE INFOCOM*, vol. 2016-July, 2016.

[8] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[9] R. Ben Basat, G. Einziger, R. Friedman, and Y. Kassner, "Optimal elephant flow detection," in *Proceedings - IEEE INFOCOM*, 2017.

[10] L. Yi, W. Mei, B. Prabhakar, and F. Bonomi, "ElephantTrap: A low cost device for identifying large flows," in *Proceedings - 15th Annual IEEE Symposium on High-Performance Interconnects, HOT Interconnects*, 2007, pp. 99–105.

[11] T. Fioreze, L. Z. Granville, R. Sadre, and A. Pras, "A Statistical Analysis of Network Parameters for the Self-management of Lambda-Connections." *Aims*, vol. 5637, pp. 15–27, 2009. [Online]. Available: http://dblp.uni-trier.de/db/conf/aims/aims2009.html#FiorezeGSP09

[12] K. C. Lan and J. Heidemann, "A measurement study of correlations of Internet flow characteristics," *Computer Networks*, vol. 50, no. 1, pp. 46–62, 2006.

[13] Y. Zhang, B. Fang, and Y. Zhang, "Identifying high-rate flows based on Bayesian single sampling," in *ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings*, vol. 1, 2010.

[14] S. Sarvotham, R. Riedi, and R. Baraniuk, "Connection-level analysis and modeling of network traffic," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 99–103.

[15] K. Psounis, A. Ghosh, B. Prabhakar, and G. Wang, "SIFT: A simple algorithm for tracking elephant flows, and taking advantage of power laws," 2005. [Online]. Available: http://www.stanford.edu/ balaji/papers/05sifta.pdf

[16] J. D. Cook, "Exact Calculation of Beta Inequalities," UT MD Anderson Cancer Center Department of Biostatistics, Houston, Texas, Tech. Rep., 2005. [Online]. Available: https://www.johndcook.com/UTMDABTR-005-05.pdf

[17] A. Papoulis, "Probability, Random Variables And Stochastic Processes," *Book*, p. 678, 1991.

[18] N. M. P. Phaal, S. Panchen, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," *IETF RFC 3176*, 2001.

[19] M. Afaq, S. Rehman, and W.-C. Song, "Large Flows Detection, Marking, and Mitigation based on sFlow Standard in SDN," *Journal of Korea Multimedia Society*, vol. 18, no. 2, pp. 189–198, 2015.

APPENDIX

A. Additional Synthetic Data Experiments

The synthetic data experiments performed when the cache size γ is equal to 40 and 200 are shown in Figures A1 and A2 respectively. Each figure shows the detection likelihood as a function of the number of packets sampled, for $\epsilon = 2, 3, 4, 5$. These figures demonstrate the trend in accuracy as the number of flows in the cache increases. The results indicate that the algorithm is fairly robust to an increased number of flows, as can be seen by comparing the iteration number of convergence (time to convergence) with the cache size γ. For example, the time to convergence for the Gaussian distribution with $\epsilon = 2$ for $\gamma = 10, 40, 100, 200$ are $599, 622, 685$, and 1723 respectively. This shows that a 20 fold increase in the number of flows led to roughly tripling the number of samples needed to converge. Similar results can be seen for different flow distributions and different values of ϵ as well.

B. Proof of Detection Likelihood Lemma

Lemma: Detection under partial information. The detection likelihood of a data network at time t follows a multivariate hypergeometric distribution as follows:

$$P(e_\epsilon(t) = 0) = P(C_\epsilon(t) = F_\epsilon) = \sum_{\forall \boldsymbol{n'} \in Z(t)} \frac{\prod_{\forall i} \binom{\sigma_i}{n'_i}}{\binom{\sum_{\forall i} \sigma_i}{\sum_{\forall i} n_i(t)}} \quad (32)$$

where Z(t) is the *zero quantum error region*, expressed as:

$$Z_\epsilon(t) = \{\boldsymbol{n'} \in \mathbb{N}^{|F|} \mid \sum_{\forall i} n'_i = \sum_{\forall i} n_i(t), \boldsymbol{n'} \leq_p \boldsymbol{\sigma},\, n'_i > n'_j \ \forall i, j \text{ s.t. } i \leq \epsilon,\, j > \epsilon\} \quad (33)$$

and $a \leq_p b$ means b is at least as Pareto efficient as a.

Proof: Assume a discrete fluid model of the network in which each flow i needs to transmit a number of water droplets equal to its size metric σ_i. Flows transmit water through the network one droplet at a time and each droplet is transmitted at arbitrary times. By convention, we will assume the first and last droplets from any of the flows are transmitted at times 0 and t_e, respectively. An observer of the network performs only one task: counting the number of droplets each flow has transmitted and storing such information in a vector $n(t)$, where each component $n_i(t)$ corresponds to the amount of droplets seen from flow i up until time t. Based on this information, the objective is to quantify the probability that the set of flows $C_\epsilon(t)$ is the same as the set of flows in F_ϵ.

At time t, the total number of droplets transmitted is $\sum_{\forall i} n_i(t)$ out of a total number of $\sum_{\forall i} \sigma_i$ droplets. The total number of possibles ways in which $\sum_{\forall i} n_i(t)$ droplets are transmitted is given by this binomial expression:

$$\binom{\sum_{\forall i} \sigma_i}{\sum_{\forall i} n_i(t)} \quad (34)$$

Only a subset of the total number of ways in which droplets are transmitted correspond to the case of zero quantum error. In particular, those vectors n' that satisfy the following conditions:

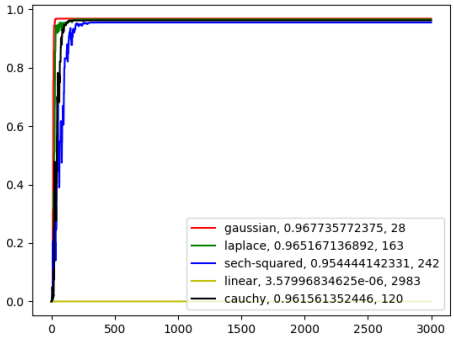
- The total number of droplets transmitted, $\sum_{\forall i} n'_i(t)$, is equal to $\sum_{\forall i} n_i(t)$.
- The number of droplets transmitted by a flow cannot be larger than its size metric: $n' \leq_p \sigma$.
- The top ϵ flows, $\{f_1, f_2, ..., f_\epsilon\}$, are captured by the set $C_\epsilon(t)$, that is, $n'_i > n'_j$ for all i and j such that $i \leq \epsilon$ and $j > \epsilon$.

The above three conditions define the zero quantum error region as expressed in Equation (33) and its cardinality is as follows:

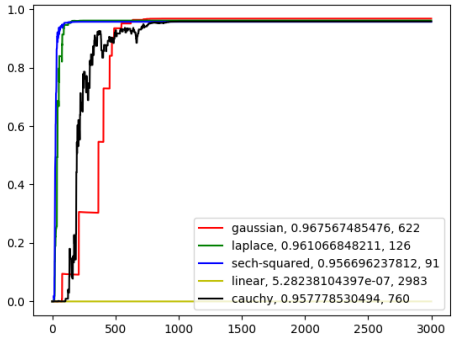
$$\sum_{\forall \boldsymbol{n'} \in Z(t)} \prod_{\forall i} \binom{\sigma_i}{n'_i} \quad (35)$$

The probability that the quantum error is zero, $P(e_\epsilon(t) = 0)$, can now be obtained from the division of Equation (34) by Equation (35).

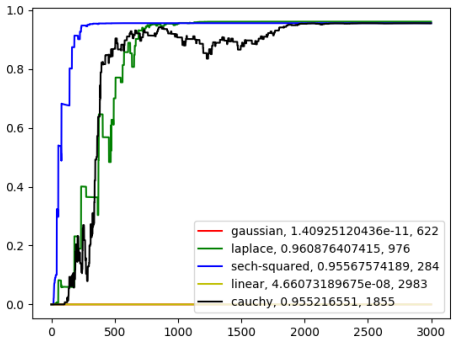
■



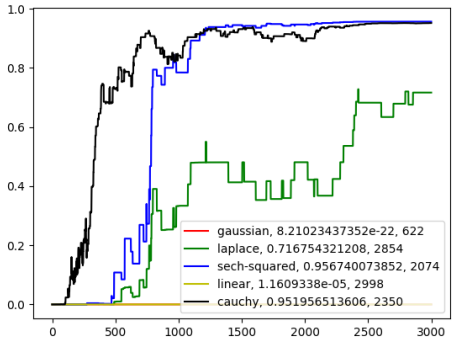
(a) $\gamma = 40$ and $\epsilon = 2$.



(b) $\gamma = 40$ and $\epsilon = 3$.

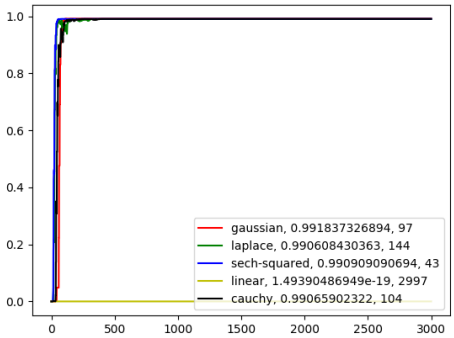


(c) $\gamma = 40$ and $\epsilon = 4$.

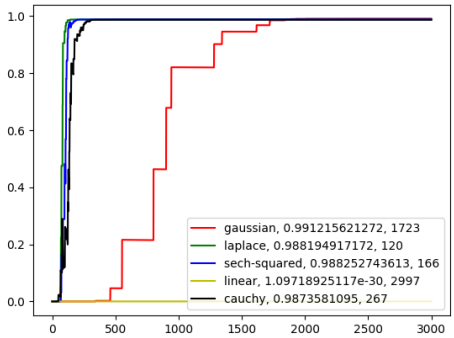


(d) $\gamma = 40$ and $\epsilon = 5$.

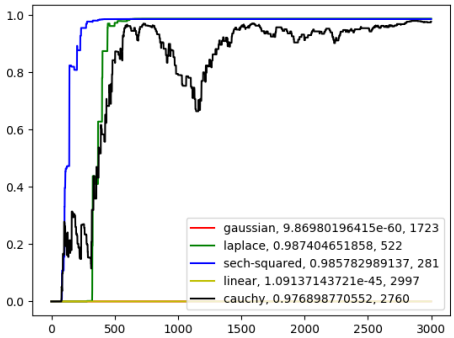
Fig. A1: Detection likelihood over the number of samples with $\gamma = 40$



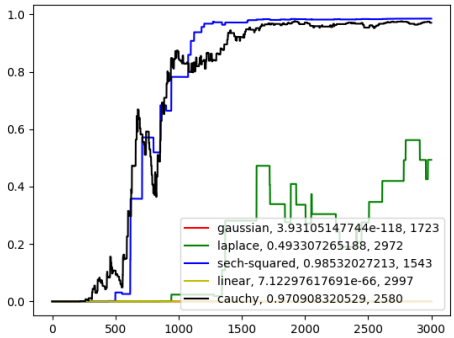
(a) $\gamma = 200$ and $\epsilon = 2$.



(b) $\gamma = 200$ and $\epsilon = 3$.



(c) $\gamma = 200$ and $\epsilon = 4$.



(d) $\gamma = 200$ and $\epsilon = 5$.

Fig. A2: Detection likelihood over the number of samples with $\gamma = 200$