Software defined Network Inference with Passive/active Evolutionary-optimal pRobing (SNIPER)

***Abstract*—A key requirement for network management is accurate and reliable network monitoring where critical information about internal characteristics or states of the network(s) must be obtained. In today's large-scale networks, this is a challenging task due to the hard constraints of network measurement resources. In this paper, a new framework (called SNIPER) is proposed where we use the flexibility provided by Software-Defined Networking (SDN) to design the optimal observation or measurement matrix which leads to the best achievable estimation accuracy using Matrix Completion (MC) techniques. Here, to cope with the inherent complexity of the process of designing large-scale optimal observation matrices, we use the well known Evolutionary Optimization Algorithms (EOA) which directly target the ultimate estimation accuracy as the optimization objective function. We evaluate the performance of SNIPER using both synthetic and real network measurement traces from different network topologies and by considering two main applications including network traffic and delay estimations. Our results show that this framework is generic and efficient that can be used for a variety of network performance measurements under hard constraints of network measurement resources. Also, to demonstrate the effectiveness and feasibility of our framework, we have implemented a prototype of SNIPER in Mininet.**

I. Introduction

Software-Defined Networking (SDN), along with its OpenFlow enabler, nicely separates the data plane and control plane functions, and provides a capability to control and reprogram the internal configurations of switches in dynamic environments. Such a flexibility can be used to adaptively and efficiently implement more complex networking applications, including both passive and active network monitoring without the need of customization. This is of particular importance in many network management and security applications where accurate and reliable network monitoring is necessary to provide critical information about internal characteristics or states of the network(s) that must be directly measured or indirectly inferred [1] [2] [3] [4].

In today's complex networks, the direct measurement of network's Internal Attributes of Interest (IAI) can be challenging or even inefficient and infeasible due to the hard constraints of network measurement resources. For example, consider the case where the IAI is the Origin-Destination Flow (ODF) sizes (as a measure of traffic intensity between nodes) or where the IAI are per-flow delay, throughput or packet loss. In large-scale networks, the measurement resources, including the Ternary Content Addressable Memory (TCAM) entries, processing power, storage capacity and limited available bandwidth, are very limited, and hence, rendering per-flow measurements are infeasible. To cope with scalability issues, Network Inference (NI) techniques can be leveraged to estimate the IAI based on a limited set of passive and/or active measurements. However, NI problems are naturally ill-posed in the sense that the number of measurements are not sufficient to uniquely and accurately determine the solution. Hence, side information from different perspectives and sources must be incorporated into the problem formulation to improve the estimation precision [5] [6] [7].

The flexibility provided by the SDN can be utilized to optimize and facilitate the process of collecting the required direct measurements and/or side information which can then be used by network inference techniques to estimate the IAI. In fact, the capabilities of SDN have been utilized in a variety of passive and active network monitoring applications. Most SDN based passive measurement studies are related to traffic engineering and network security applications, such as, network traffic measurement or identifying Heavy Hitters (HH) and Hierarchical Heavy Hitters (HHH). In [1] and [2], SDN reconfigurable measurement architectures are proposed where a variety of sketches for different direct measurement tasks can be defined and installed by the operator. In [8], OpenTM directly measures a traffic matrix by keeping track of statistics for each flow. Recently, in [3], an intelligent SDN based traffic measurement framework (called iSTAMP) with the ability of adaptive and accurate fine-grained flow estimation is proposed. For active network measurement under SDN paradigm, the very recent work [4] establishes a general framework (called Opennetmon) where accurate measurements of per-flow throughput, packet loss and delay can be directly measured.

However, under hard network resource constraints, the state of the art SDN-enabled traffic measurement and inference methods, for example [2] and [3], suffer from the following challenges. First, the application of these frameworks are mainly limited to network traffic measurement. Second, the longest prefix matching forwarding in OpenFlow implies that incoming flows can be aggregated in just one entry of the TCAM, and hence, the capability of providing optimal redundant aggregated measurements is limited. Third, in [3], to simplify the process of designing the optimal aggregation (i.e. measurement) matrix, the ultimate estimation accuracy is not directly targeted; instead, the coherency of the measurement matrix is minimized, leading to unavoidable sacrifice in the performance [3][9].

On the other hand, recently, Matrix Completion (MC) techniques have been used as powerful network inference tools that involve completing a matrix of IAI from the direct measurement of a sub-set of its independent entries [10][11][12]. Examples of the matrix of IAI include a matrix where each entry is an ODF at different times [10], or per-flow delay/packet-loss between different nodes of the network [12]. Since a variety of resources and information are often shared across different layers in communication networks, the main assumption in MC techniques is that the matrix of IAI is a low-rank matrix which contains spatio-temporal redundancies, and thus, not all of its entries are needed to represent it; accordingly missed or non-observed entries can be estimated from a sub-set of randomly measured entries. In the theory of matrix completion, the matrix of IAI can be completely reconstructed from a sub-set of observed/measured entries (indicating the observation/measurement matrix) if the number of *randomly* chosen observations are *high* enough [13][14]. Accordingly, in [10] and [11] the MC methods are used for network Traffic Matrix (TM) completion to estimate the missed entries of the TMs. Also, in [12], a new MC technique has been used for active network performance measurements where the status of path delays or bandwidths are predicted from a set of active measurements and using a new MC technique.

The flexibility provided by the SDN coupled with the capability of MC techniques, in reconstructing the matrix of IAI from a sub-set of directly measured *independent* entries, pave the way for: 1) designing an efficient framework for different passive/active network measurement applications under hard constraint of measurement resources, and 2) providing required side information without feasibility constraints (as in [3]). In this paper, we use different matrix completion techniques as our main NI tools, and define an observation matrix as a matrix which provides required direct measurements of IAI that can be used by MC algorithms. Specifically, we answer the following interesting question:

Under hard network resource constraints, how can we use the SDN capabilities to measure a sub-set of entries of the matrix of IAI and design the optimal observation matrix which leads to the best possible estimation accuracy using matrix completion techniques?

However, the *direct* design of optimal observation matrices for maximizing the performance of NI methods is prohibitive due to the complexity of the process [3][9]. The underlying difficulty lies in the fact that formulating the network inference process or algorithm, as a function of the observation matrix which targets the ultimate estimation accuracy, into a closed-form and well-defined optimization problem that can be efficiently optimized is extremely complicated and computationally complex, if it is not impossible or intractable. Therefore, in this paper, we propose a new approach in designing the optimal observation matrix for network inference problems where we *directly* target the ultimate estimation accuracy in network monitoring applications in our optimization framework. However, to cope with the inherent complexity of the process of designing large-scale optimal observation matrices, we use the well known Evolutionary Optimization Algorithms (EOA) that are suitable for the optimization problems where the main objective function is a procedure or an algorithm that can not be formulated as a well-defined mathematical function. In this framework, the evolutionary optimization algorithm acts as a *sniper* which precisely captures or measures the best or the most informative entries of the matrix of IAI which leads to the best estimation accuracy via using matrix completion techniques. We refer to our proposed framework as Software defined Network Inference with Passive/active Evolutionary-optimal pRobing (SNIPER). The SNIPER is a simple, flexible, and efficient framework which can be easily deployed on commodity OpenFlow-enabled routers/switches to enhance the performance of various passive or active network monitoring applications with low computation and communication overhead between control and data planes. Since MC techniques (as the main NI methods employed by SNIPER) directly use partial independent measurements of IAI, this framework can be easily implemented in a centralized or distributed manner. Accordingly, it is compatible with the recent trends in developing more smart and agile SDN platforms [15][16] where data plane APIs and switches are able to execute codes inside the device with no further interaction with the controller.

Our main contributions are summarized as follows:

• To the best of our knowledge, this is the first time that EOAs are applied to design the optimal observation matrix where ultimate network inference performance is the main objective function to be optimized. We show that under hard constraint of measurement resources the optimal design of the observation matrix provides more accurate estimates in different network monitoring applications, including per-flow size and delay estimations.

• We address the scalability, deployability and feasibility of the SNIPER framework: a) by reducing the computational complexity of EOAs; b) by introducing a new adaptive algorithm for network measurement in dynamic environments, and c) by evaluating the performance of our framework using both synthetic and real network measurement traces from different network topologies. We applied SNIPER to two main applications: network traffic and delay estimations. In addition, we implemented a prototype of SNIPER in Mininet.

The rest of this paper is organized as follows. Section II provides an overview of the SNIPER framework and the matrix completion techniques that we have used as our main NI methods. In Section III we describes our optimal observation matrix design procedure using the EOAs. Then, in Section IV, we explain our methodology for evaluating the performance of the SNIPER. In Section V, we evaluate the performance of SNIPER considering two main applications including per-flow path delay and per-flow size estimations. Section VI summarizes our most important results.

II. SNIPER: System Description

Figure 1 shows the general block diagram of the SNIPER framework where the controller interacts with Software Defined Measurement Network (SDMN) via Network Management (NM) and Optimal Network Measurement Probing (ONMP) messages to reconfigure the SDMN and poll the required measurements and statistics of the operating network. The SDMN consists of a sub-set of OpenFlow Switches (OFS) in the operating network which guarantees all required IAI are *observable* and *measurable* using the SDMN, as it is our assumption here. The NM messages include regular network operating and control commands while ONMP messages include passive/active probing messages that indicate which IAI must be accurately measured at different times and/or spaces. In the SNIPER framework, the network measurement process is consisted of two stages, namely the learning and

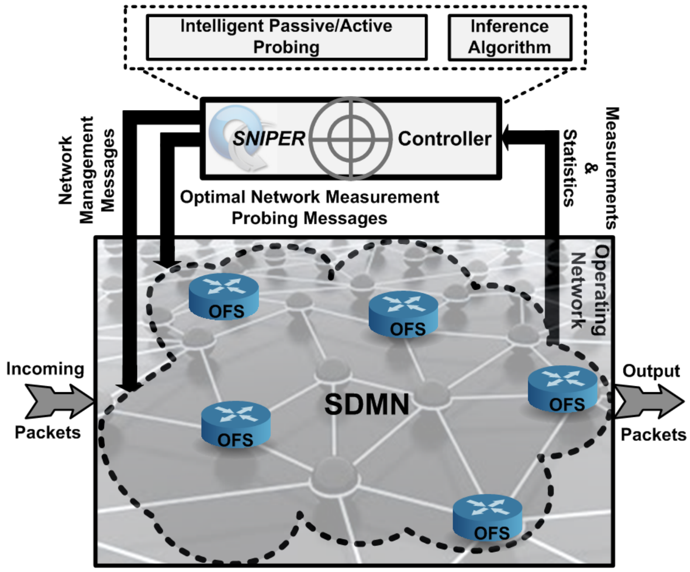

Fig. 1: SNIPER network measurement framework: a general perspective.

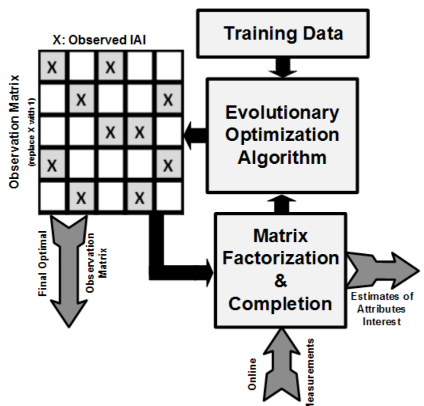

Fig. 2: Evolutionary optimal network probing process.

measurement epochs, as it is shown in Figure 2. In the learning stage, the ONMP messages are designed and computed using a supervised learning scheme where an evolutionary optimization algorithm uses a training data set to precisely design the ONMPs. Then, in the online measurement epoch, the flexibility provided by the SDN is used to reconfigure the SDMN and to collect the measurements of corresponding ONMPs. By decreasing the dependency of the SNIPER framework on the initial training data set and adaptively updating the optimal observation matrix, this framework can also be deployed in dynamic environments.

The SNIPER controller can both preconfigure or adaptively reconfigure the flow-tables of the OFSs in SDMN. For passive per-flow size measurement, the flow-tables of the SDMN is configured and per-flow counter statistics are measured. However, for active measurements (e.g. per-flow delay/loss/throughput) an appropriate path is first determined and the flow-tables of the SDMN is adaptively reconfigured; then, probing packets are injected into the network, and required IAI are measured. Accordingly, the SNIPER architecture can obtain information of active flows and monitor the end-to-end network performance measurements. These measurements are transmitted to the SNIPER controller where matrix completion techniques are used as the main NI algorithm to estimate all unknown IAI. In this framework, the communication overhead between switches and controller is low, since MC techniques only needs partial measurements.

The ONMPs are consisted of passive and active probes. Passive probes are measurement massages that reconfigure the OFSs of the SDMN by installing different rules in flow tables and defining the required actions on the incoming packets of the operating network. On the other hand, active probes not only install the rules of flow table(s) and define their actions but also use part of the network resources to inject predefined network measurement probes which will be captured by reconfigured OFSs and provide corresponding statistics. The feasibility of such Software-Defined Measurement (SDM) frameworks have been independently investigated in [3] and [4] where the capability of OpenFlow switches can be effectively utilized to measure and inference the IAI of the operating network. Likewise, the SNIPER controller is capable of providing: 1) per-flow sizes [3] by installing the flow ID prefixes in the flow tables and polls the statistics; 2) per-flow throughput [4] by determining specific path for each flow and queries switches to retrieve per-flow statistics where each query determines the amount of bytes sent and the duration of each flow; 3) per-flow packet loss [4] by polling flow statistics from the first and last switch of each path and subtracting the increase of the source switch packet counter with the increase of the packet counter of the destination switch, and 4) per-flow delay [4] by assigning a specific path to each flow and regularly injecting packets into the first switch and having the last switch send them back to the controller where the difference between the packet's departure and arrival times are computed by subtracting the estimated latency from the switch-to-controller delays. In this paper, the performance of the SNIPER framework is mainly examined in two main applications: per-flow size and delay estimations.

A. SNIPER: Problem Statement

Network monitoring is the problem of inferring the IAI corresponding to the performance of the operating network. The operating network is modeled as a connected undirected graph $G(V, E)$ where $|V| = N$, and $|E| = m$. Accordingly, there are M links, and $n = N(N-1)$ paths and flows in the network, assuming that there exists an unique path between any pair of nodes in the network. As it was explained in the introduction, in the SNIPER framework the NI problem is modeled as a Matrix Completion (MC) problem where the problem is the completion of a matrix of attributes of interest (X) from the direct measurement of a sub-set of its entries assuming that X is a low-rank matrix which contains redundancies and thus not all of its entries are needed to represent it. Here, X is a matrix of size $n \times \mathcal{T}$ ($\mathcal{T} < n$) with rank $r << \mathcal{T}$ where K entries of X is directly measured.

The theory of matrix completion [13] shows that under some suitable conditions, with high probability, X can be exactly recovered from a set of sufficient *randomly* observed entries. In practice, X is often full rank but with a rank r dominant component, that is, X has only r significant singular values σ_1,..., σ_r (where $\sigma_1 \leq ... \leq \sigma_r$) and the others are negligible. In such cases, by minimizing the rank, a matrix of rank r (denoted by $\hat{X}$) can still be found that approximates X with high accuracy [12][13][17]. Since direct minimization of the rank of a matrix is difficult, MC problems is often formulated as a convex optimization problem Eq.(1) where Ω is the set of observed (i.e. directly measured) entries, P_Ω is a sampling function that preserves entries of X in Ω (i.e. $[P_\Omega(X)]_{ij} = x_{ij}$) and turns out the others into zero, and $L(X, \hat{X}) := \sum_{i,j=1}^{n}(x_{ij} - \hat{x}_{ij})^2$. Corresponding to the sampling function P_Ω, a Binary Observation Matrix S_Ω is also defined where $[S_\Omega(X)]_{ij} = 1$. Accordingly, the MC searches for a low-rank matrix $\hat{X}$ that approximates X with sufficient accuracy at the observed entries in Ω. The unobserved or missing entries in X (indicated by $\bar{\Omega}$ as the complement of Ω) are predicted by the corresponding entries in $\hat{X}$. The MC problem can also be reformulated as a matrix factorization problem in Eq.(2) where $\hat{X}$ (with $rank(\hat{X}) \leq r$) is factorized as $\hat{X} = U_{n\times r}V_{r\times n}^T$ and λ is the regularization coefficient that controls the extent of regularization. Here, the Frobenius norm of a matrix Z is defined as $\|Z\|_F^2 = \sum_{i,j=1}^{n} |z_{ij}|^2$.

$$\begin{aligned} &\text{minimize} \quad Trace\left(\hat{X}\right) = \sum_{i=1}^{n} \sigma_i \\ &\text{s.t.} \quad L\left(P_\Omega(X), P_\Omega(\hat{X})\right) \leq \delta \end{aligned} \tag{1}$$

$$\underset{U,V}{\text{minimize}} \quad L\left(P_\Omega(X), P_\Omega(\hat{X})\right) + \lambda(\|U\|_F^2 + \|V\|_F^2) \tag{2}$$

The optimization problem Eq.(2) can be solved using different methods. In this paper, we adopt two different methods from recently MC procedures used in network monitoring applications to solve Eq.(2) and compute U and V matrices where $\hat{X} = UV^T$. The first one is the Sparsity Regularized Singular Value Decomposition (SRSVD) method [10] that uses an alternating least squares procedure to solve Eq.(2). The second one is the Decentralized Matrix Factorization algorithm [12],denoted by DMFSGD, that uses the Stochastic Gradient Descent (SGD) technique to solve Eq.(2). Both methods rely on the fact that the matrices of IAI in network monitoring applications contain temporal and/or spatial redundancies that can be used to estimate non-observed or missed entries.

Under hard resource constraints, it is crucial to design the optimal observation matrix, which leads to the best achievable estimation accuracy using matrix completion techniques. To show the importance of such a design, consider a 3×3 matrix X consisting of three spatial-independent processes in each row where $x_1(t) = \frac{1}{2}(x_1(t-1) + x_1(t+1))$, $x_2(t) = 2x_2(t-1) + 3$ and $x_3(t) = \frac{1}{2}x_3(t+1) - 10$. Using the temporal structure in these processes, an optimal observation matrix can be designed as $\Omega_{Opt} = [1, 0, 1; 1, 0, 0; 0, 0, 1]$ where there is at least one 1 in each row. Note that such an optimal observation matrix can not guaranteed to be obtained using a random sampling strategy.

To maximize the performance of MC algorithms with minimum number of required measurements, such the process of designing the optimal observation matrix must directly target the ultimate estimation accuracy in the network monitoring applications as defined in Eq.(6). However, it is extremely complicated, if it is not impossible, to formulate the MC process and target the ultimate performance criterion using a closed-form and well-defined mathematical optimization problem as a function of the observation matrix. In addition, since in our applications the observation matrix is a binary matrix, it is computationally expensive and intractable to use integer optimization techniques in such a design process for large-scale networks. Therefore, in this paper, to cope with the inherent complexity of the process of designing large-scale optimal observation matrices, we use the well known evolutionary optimization algorithms that are suitable for the optimization problems where the main objective function is a procedure or an algorithm.

III. Evolutionary-Optimal Observation Matrix Design

Evolutionary algorithms are the sub-category of heuristic optimization methods [18] for solving NP-hard optimization problems where the main objective function may not be formulated as a well-defined mathematical function. As Figure 3 shows, evolutionary algorithms consists of three main processes. The first process is the initialization process where the initial population of individuals is randomly generated according to some solution representation. Each individual represents a solution. In the second process, each solution in the population is then evaluated for a fintness value. The fitness values can be used to calculate the average population for the purpose of selection. The third process is the generation of a new population by the perturbation of solutions in the existing population. The algorithm is run until the stopping criterion is met. In this paper, we use two EOAs including Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) which have been applied to many optimization and machine learning problems [18][19].

The main idea of GA is to mimic the natural selection mechanism and the survival of the fittest. In GA, the solutions are represented as chromosomes. The chromosomes are evaluated for fitness values and they are ranked from the best to the worst based on fitness value. The process to produce new solutions in GA is accomplished through three genetic operators as selection, crossover, and mutation. First, the better chromosomes are selected as parents to generate new offspring (new chromosomes). To simulate the survivor of the fittest, the chromosomes with better fitness are selected with higher probabilities. Once the parent chromosomes are selected, the crossover operator combines the chromosomes of the parents to produce new offspring (perturbation of old solutions). To avoid stagnation in the process of evolution, the mutation operator is performed on the chromosomes to increase the diversity of the population. To successfully apply the GA, the solution representation (i.e. chromosome model) must be designed carefully. Also, the parent selection process, and the probability of crossover and mutation are important parameters that must be precisely chosen [19].

In PSO, a solution is represented as a particle, and the population of solutions is called a swarm of particles. The first process in PSO is the initialization process where the initial swarm of particles is generated. Each particle is initialized with a random position and velocity. Each particle is then evaluated for fitness value. Each time a fitness value is calculated, it is compared against the previous best fitness value of the particle and the previous best fitness value of the whole swarm, and, accordingly, the personal best and global best positions are updated, appropriately. If a stopping criterion is not met, the velocity and position are updated to create a new swarm. The positions and velocities of particles are updated based on the personal best and global best positions, as well as the old velocities. It should be noted that PSO algorithm does not require sorting of fitness values of solutions in any process. This might be a significant computational advantage over GA, especially when the population size is large [19] [20]. Here, a solution in the GA is represented as a chromosome which is defined as a binary sampling matrix C with size $n \times \mathcal{T}$ and where 0 and 1 respectively represent unobserved and directly measured entries. The number of measurements paths (i.e. samples) for each chromosome is denoted by K (i.e. the number of one's in each chromosome). To successfully apply the MC technique, the sampling matrix C is constrained to have at least one 1 in each row and column. The GA is started by generating N_p chromosomes/solutions in the initialization step and estimating all unknown IAI in the set $\bar{\Omega}$ using MC algorithm. Then, the fitness of each chromosome is evaluated using the cost function Eq.(6). Accordingly, the best chromosomes, with lowest fitness values, are selected and the crossover operation, with probability p_c, is applied on each pair of parents to generate new children (offsprings). Eq.(3) defines the crossover operation where r_c denotes a randomly chosen row from set $\{1, ..., n\}$. These Offsprings form part of the new chromosomes of the next generation. To increase the diversity of the population, the mutation operation is performed on each child where the mutation operator changes an entry of sampling matrix C from zero-to-one or vice-versa with probability p_m. The GA process is continued over N_i iterations and the best chromosome in each iteration remains unchanged. In most cases throughout this paper, the GA parameters are set as: $N_i = 60$, N_p=1500, $p_c = 0.3$, and $p_m = 0.01$.

$$\begin{aligned} \text{OffSpring}_1 &= C_1(1:r,:) + C_2(r+1:n,:) \\ \text{OffSpring}_2 &= C_2(1:r,:) + C_1(r+1:n,:) \end{aligned} \tag{3}$$

Likewise, the PSO is started by generating N_p particles and estimating all unknown IAI in the set $\bar{\Omega}$ using the MC algorithm. The i^{th} particle is identified by its position P_i^k and its velocity V_i^k at iteration k. Here, P_i^k is an $n \times \mathcal{T}$ binary matrix, representing the measurement matrix, and V_i^k is also an $n \times \mathcal{T}$ matrix. In the initialization stage all position and velocity matrices are zero matrices. The best position of i^{th} particle obtained until iteration k is denoted by BP_i^k and the best position among all particles in the swarm until iteration k is called global best position and it is denoted by GP^k. The best particles here is determined by evaluating the fitness of each particle and choosing the one with the minimum error value (as defined in Eq.(6)) among all iterations (for one particle) or among all particles. The velocity V_i^k is updated according to Eq.(4) where β_1 and β_2 are acceleration constants, which here they setup to $\beta_1 = \beta_2 = 2$, and α_1 and α_2 are standard uniform random variables in interval $[0, 1]$. The positive inertia weight ω is computed as $\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{k}{N_i}$ where ω_{min} and ω_{max} are respectively minimum and maximum inertia weights which, here, we setup to $\omega_{min} = 0.3$, $\omega_{max} = 0.9$ and $N_i = 2000$. The particle positions are updated (by re-determining new IAI) using two methods: 1) set the entries $\{p_{i_{kl}}^k\}_{kl \in I_{max}^V} = 1$ where I_{max}^V indicates the set of kl^{th} entries with highest velocities in the matrix V_i^k (i.e. $(\sim, I_{max}^V = sort(abs(V_i^k(:))))$, and 2) $\{p_{kl}\}$ is set to one with probability $sigmoid(v_{ij})$, where $sigmoid(x) := \frac{1}{1+e^{-x}}$, otherwise it is set to zero. The PSO process is continued for N_i iterations.

$$V_i^k = \omega V_i^{k-1} + \alpha_1\beta_1(BP_i^k - P_i^{k-1}) + \alpha_2\beta_2(GP^k - P_i^{k-1}) \tag{4}$$

In both GA and PSO evolutionary algorithms, simple manipulations are applied at each step to keep the number of observed IAI constant for each sampling rate in such a way that chromosomes/particles remain symmetric (if it is required, e.g., in the case of delay measurement) with having at least an one in each row and column of the solution representation.

IV. SNIPER Performance Evaluation Methodology

The performance of SNIPER is evaluated in two main applications, including per-flow size and per-flow delay estimations via matrix completion. For this purpose three network topologies, including Abilene, Geant and Harvard networks, and both synthesis and real network traces are considered. For per-flow size estimation, we use real traffic traces from Abilene [21] and GEANT [22] networks; the characteristics of these traffic traces are represented in Table I. For per-flow path delay estimation, we first use the Abilene and Geant network topologies to generate the required synthetic data set where it is assumed that the path delay for the flow between node i and node j is modeled as Eq.(5). In this model, d_{ij}^p is the propagation delay between i^{th} and j^{th} nodes, and q_{ij} is the queuing delay in which ,according to [23], it is modeled as $q_{ij} \sim exp(\lambda)$. Since the average propagation delay in both Abilene and Geant networks is approximately 3.5 ms, thus, the range of the variation of λ is chosen in $0 \leq \lambda \leq 10$ which includes both low and high noise scenarios. In addition, we use real per-flow delay from Harvard [24] which contains 2,492,546 measurements of application-level RTTs, with timestamps, between 226 Azureus clients collected in 4 hours [12].

$$d_{ij} = d_{ij}^p + q_{ij} \tag{5}$$

In our supervised learning scheme, each data set is divided into t_p parts. The first part, called learning epoch, with size $n \times T_0$ (where $T_0 = \left\lceil \frac{\mathcal{T}_0}{t_p} \right\rceil$) is utilized to design the ONMP messages using the GA and PSO evolutionary algorithms. The last population of the learning stage determines the ultimate ONMP and its estimation performance is denoted by lower script T_0 in our results. Then, the same ONMP set is used over other $t_p - 1$ parts of the data set (called measurement epochs) and the average of the performance over multiple

Network	Date	Duration	Resolution	TM Size ($n \times \mathcal{T}$)
Abilene [21]	2004-05-01	1 week	5 min.	144 × 2016
GEANT [22]	2005-01-08	1 week	15 min.	529 × 672

TABLE I: Real Datasets under study.

parts is computed and is denoted by lower script Avg in our results. The number of measurement paths, denote by K, plays an important role in improving the estimation accuracy. This parameter is defined as $K = s.(n.\mathcal{T})$ where s is the Sampling Ratio (SR) and $0 \leq s \leq 1$. Note that, the higher the K is, the better the estimation accuracy is.

The performance of NI methods in SNIPER framework, that is, the estimation accuracy of the completion of the matrix of IAI is evaluated using the following two criteria in Eq.(6) where NMAE denotes Normalized Mean Absolute Error and NMSE denotes Normalized Mean Square Error. The status of the IAI are also classified into two different classes. In the case of classifying per-flow delays, the flow delay estimates are compared with a threshold θ which is set as the average delay in the data set. On the other hand, in the case of classifying per-flow sizes, the flow size estimates are compared with a threshold θ which is set as a fraction of the link capacity C_l; here, C_l is set to the maximum flow size in the available data set. Accordingly, the performance of the detection of congested paths (i.e. flows with delay longer than the threshold) and heavy hitters (i.e. flows larger than the threshold) are computed by the probability of detection P^d and probability of false alarm P^{fa} in Eq.(7). Here, different lower scripts are used to distinguish between different applications where CP denotes Congested Paths and HH denotes Heavy Hitters, respectively.

$$
\begin{aligned}
NMAE &= \frac{\sum_{ij \in \bar{\Omega}} |x_{ij} - \hat{x}_{ij}|}{\sum_{ij \in \bar{\Omega}} |x_{ij}|} \\
NMSE &= \frac{\sqrt{\sum_{ij \in \bar{\Omega}} (x_{ij} - \hat{x}_{ij})^2}}{\sqrt{\sum_{ij \in \bar{\Omega}} (x_{ij})^2}}
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
P^d &= \frac{1}{|\bar{\Omega}|} \sum_{ij \in \bar{\Omega}} Pr\left(\hat{x}_{ij} \geq \theta | x_{ij} \geq \theta\right) \\
P^{fa} &= \frac{1}{|\bar{\Omega}|} \sum_{ij \in \bar{\Omega}} Pr\left(\hat{x}_{ij} \geq \theta | x_{ij} < \theta\right)
\end{aligned} \tag{7}
$$

V. The Applications of SNIPER Framework

In this section, the effectiveness of our network measurement and inference framework in Figure 1 is justified for two main applications, including per-flow delay and size estimations, and under different configurations. Each configuration determines the network under study, the matrix completion technique, the length of learning period T_0, and the sampling ratio s. Here, T is set to $T = 100$; however, s mainly varies in the range of small values to indicate a case of hard constraint of network measurement resources. Other parameters, such as the number of measurement paths K and the number of parts in the data set t_p can be determined, accordingly. The type of sampling strategies are denoted by RS, GA and PSO which respectively identify the ONMP designed by Random Sampling (RS) and evolutionary algorithms GA or PSO. Note that, the performance of RS strategy is evaluated using Monte-Carlo simulation with 100 iterations.

A. Optimal Observation Matrix Design using SNIPER

The optimal design of large-scale binary observation matrices, using mathematical optimization techniques are extremely complicated or computationally expensive. Here, to show the effectiveness of our evolutionary optimal observation matrix design, we consider a small ring network consisting of 4 nodes with different per-flow path delays where we can compute all possible observation matrices at a specific sampling ratio. Then, we estimate the unobserved entries and compute the corresponding NMSE for all possible observation matrices. Using this process we realized that our EOAs are able to obtain the optimal observation matrix. As an example, if $X = [0, 5.05, 9.01, 9.645; 5.05, 0, 3.96, 9.645; 9.01, 3.96, 0, 5.23; 4.595, 9.645, 5.23, 0]$ (in ms) and $s = 0.25$, then the optimal observation matrix $S_\Omega^{Opt}(X)$ with $NMSE = 0.4734$ is $S_\Omega^{Opt}(X) = [0, 0, 1, 0; 0, 0, 0, 1; 1, 0, 0, 0; 0, 1, 0, 0]$ which is also obtained by our GA in SNIPER framework.

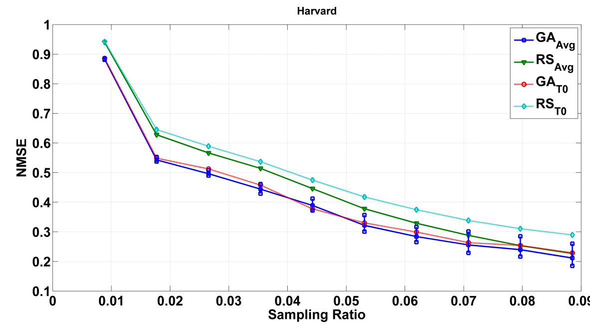

Fig. 6: The $NMSE$ for Harvard network in different sampling ratios.

B. Per-Flow Delay Estimation using SNIPER

Figures 4 and 5 show the performance of the SNIPER in the estimation of per-flow delay on Abilene and Geant networks using synthesis data generated using the model in Eq.(5) where the MC technique is DMFSGD as in [12]. Here, the ONMP is designed using the GA and only by considering the propagation delay in Eq.(5) in the learning epoch. Then, this ONMP is used to evaluate the performance of the MC technique in measurement epochs, based on the $NMSE_{Avg}$, where queuing delay is added to the propagation delay as Eq.(5) models the network paths delay [23]. Figure 6 also shows the performance of the SNIPER framework on real per-flow delay from Harvard [24] network. In this figure, and throughout this paper, the blue squares represent the minimum and maximum of $NMSE$ or (or $NMAE$) for each sampling ratio using our EOA based sampling strategy. It should be noted that, in low sampling ratios and in all measurement epochs a better estimation accuracy is obtained comparing with random sampling strategy. In addition, Table II indicates the capability of the SNIPER framework in the reliable detection of Congested Paths (CP). These results show that, at lower SRs which indicates the hard resource constraint regime, by the intelligent design of the ONMP messages using the SNIPER framework a better estimation accuracy, with more robust performance against noise, can be obtained via applying MC techniques. This is an important factor in active network performance measurement, including delay, where the communication bandwidth is the main resource for accurate measurement and it is very limited.

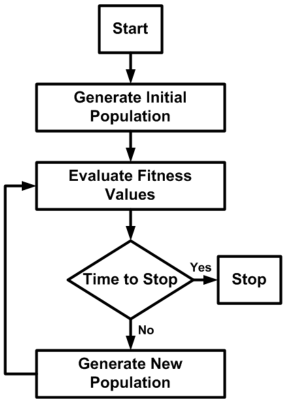

Fig. 3: The flowchart of evolutionary optimization algorithms.

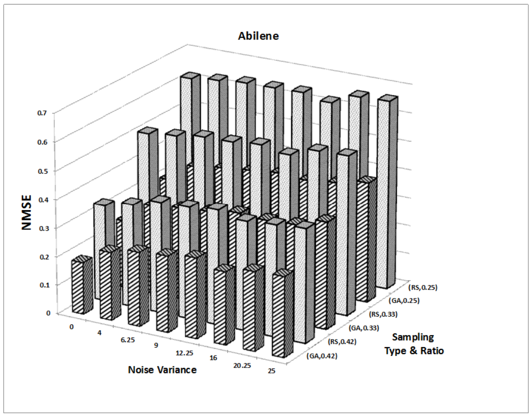

Fig. 4: The $NMSE$ v.s. SR & noise for Abilene.

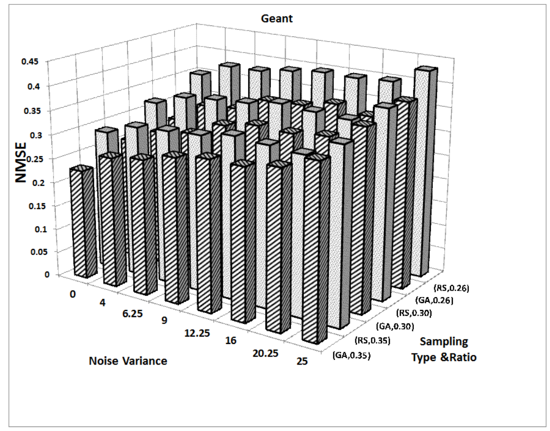

Fig. 5: The $NMSE$ v.s. SR & noise for Geant.

SR	0.0088	0.0265	0.0442	0.0619	0.0796
P_{CP}^{d}	0.8077	0.9135	0.9306	0.9499	0.9566
P_{CP}^{fa}	0.4749	0.2412	0.1661	0.1263	0.1032

TABLE II: Average P_{CP}^{d} and P_{CP}^{fa} for Harvard network.

	$SR=0.2$	$SR=0.3$	$SR=0.4$	$SR=0.5$
P_{HH}^{d} Abilene (RS)	0.6256	0.7544	0.8426	0.8851
P_{HH}^{d} Abilene (GA)	0.6550	0.7693	0.8380	0.8901
P_{HH}^{fa} Abilene (RS)	0.0353	0.0202	0.0144	0.0116
P_{HH}^{fa} Abilene (GA)	0.0325	0.0192	0.0142	0.0119
P_{HH}^{d} Geant (RS)	0.7606	0.8935	0.9354	0.9489
P_{HH}^{d} Geant (GA)	0.7804	0.9096	0.9375	0.9502
P_{HH}^{fa} Geant (RS)	0.0106	0.0061	0.0043	0.0035
P_{HH}^{fa} Geant (GA)	0.0095	0.0058	0.0041	0.0034

TABLE III: Comparing the average P_{HH}^{d} and P_{HH}^{fa} between RS and GA sampling and for Abilene and Geant networks where $\theta = 0.05C_l$ and $\theta = 0.1C_l$, respectively.

C. Per-Flow Size Estimation using SNIPER

Figure 7 shows the performance of the SNIPER in the estimation of per-flow sizes on both Abilene and Geant networks using real traffic traces (see Table I) where the MC technique is the SRSVD as in [10]. Again, it is clear that by the optimal design of network measurement probes or equivalently the observation matrix, the performance of the matrix completion is improved, particularly at low sampling ratios which indicates the hard resource constraint of TCAM entries as the main resource for per-flow size measurement. The better accuracy is obtained almost for all sampling ratios. Table III also shows the average performance of the SNIPER framework in the reliable detection of heavy hitters under low sampling ratios.

D. Scalability of SNIPER

As we have seen in the previous results, the SNIPER can improve the estimation accuracy under hard resource constraint regimes. To reduce the high computational complexity of the GA in designing the optimal observation matrix in large-scale networks and increase the scalability of the SNIPER framework, here, we use the PSO evolutionary optimization algorithm which is much faster than the GA [18][19] and it can reduce the computational complexity and processing power of the SNIPER. Figure 8 shows the performance of SNIPER for per-flow size estimation, representing the fact that in low sampling rates the intelligent design of the observation matrix using PSO algorithm results in a better estimation accuracy. The reduction in the computational complexity using the PSO algorithm is quantified using the notion of Processing Gain (PG) defined as $PG := 100 \times \frac{PT_{GA} - PT_{PSO}}{PT_{GA}}$ where PT_{GA} and PT_{PSO} respectively denote the processing times for running GA and PSO algorithms. Accordingly, the processing gains for Abilene and Geant networks are PG=56% and PG=65%, respectively.

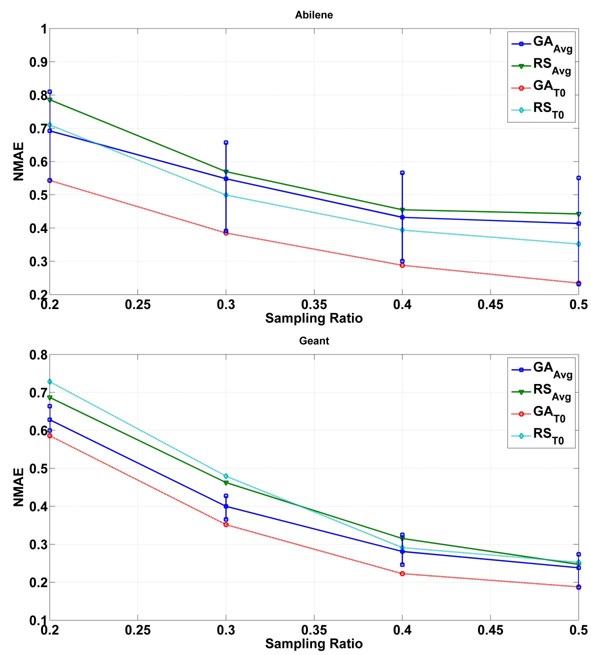

Fig. 7: $NMAE$ v.s. sampling ratios.

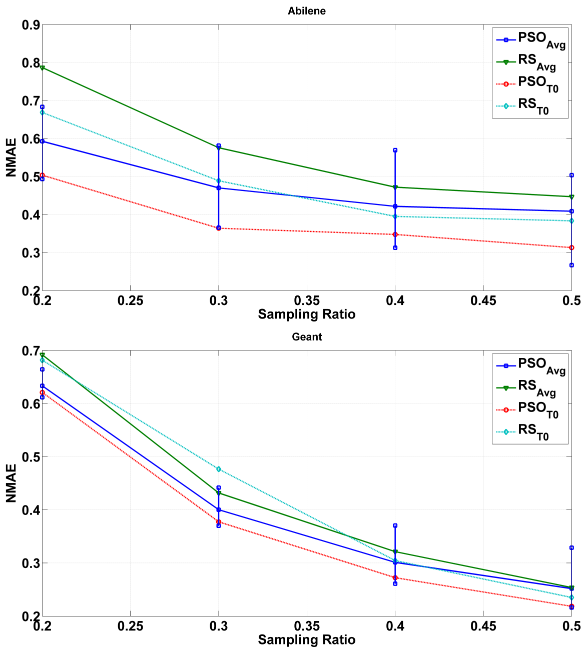

Fig. 8: $NMAE$ v.s. sampling ratios.

E. Deployability of SNIPER in Dynamic Environments

In the case of supervised learning, the SNIPER framework computes the optimal sampling matrix using the training data, available in the initial learning stage. The training data set can be obtained by directly measuring the required IAI in the beginning or by using already available data sets (e.g. NetFlow records in the case of TM completion). Under hard constraint of network measurement resources, to effectively apply the SNIPER framework in dynamic environments, it is important to: 1) decrease the dependency of the SNIPER framework on the initial training data set, and 2) adaptively update the optimal observation matrix designed in the initial stage based on the most current behavior of the network under study. Accordingly, here, we propose a new algorithm that can be effectively utilized for network measurement purposes in the SNIPER framework.

In this algorithm, to reduce the dependency of the SNIPER framework on the initial training data set, first, we determine the initial sampling ratio (indicated by s_0) and randomly measure a sub-set of IAI to form a matrix with size $n \times T_0$. Then, we apply the appropriate matrix completion on this matrix of IAI to estimate unknown IAI and form our new training data set. Then we apply the evolutionary optimization algorithm (e.g. GA) on this new training data set to compute the optimal observation matrix. By applying the MC algorithm using this optimal observation matrix we can estimate all unknown IAI. Since the largest IAI are potentially the most informative ones for increasing the estimation accuracy [3], to adaptively update the optimal observation matrix for use in the next measurement epoch, we also choose a small number of largest estimated IAI (indicated by k_0) to be measured directly. That is, after learning epoch where the optimal observation matrix is designed using the new training data set, we modify this optimal measurement matrix by adding k_0 direct measurements which are indicated by the largest estimated IAI in the previous epoch.

SR	RS_{T_0}	GA_{T_0}	GA_{Avg}	RS_{Avg}	DRS_{Avg}	DG
Abilene (s_0=0.5)						
Abilene (s_0=0.6)						
Abilene (s_0=0.7)						
Geant (s_0=0.5)						
Geant (s_0=0.6)						
Geant (s_0=0.7)						

TABLE IV: $NMAE$ for different s_0 where s=0.2 and $k_0 = 5$.

SR	0.2	0.3	0.4	0.5
GA_{T_0}	0.7739	0.6646	0.5380	0.4341
RS_{T_0}	0.8354	0.6667	0.5561	0.4518
GA_{Avg}	0.7946	0.6422	0.5172	0.4272
RS_{Avg}	0.8612	0.6587	0.5323	0.4377

TABLE V: The $NMAE$ via implementing SNIPER for Geant Network in Mininet.

Table IV shows the performance of this algorithm where the genetic algorithm is used to design the optimal observation matrix. Here, the notions of DGA and DRS are used to respectively denote Dynamic GA and Dynamic RS indicating the case where initial observation matrix is adaptively updated. It is clear that, under hard resource regime, that is at low sampling ratio SR=0.2, the DGA is able to provide more accurate estimates of estimated flows on both Abilene and Geant networks. Also, there is a trade-off between the performance and parameters s_0, which controls the dependency of the SNIPER framework on the training data set, and k_0 which determines the number of new measurements to update the optimal observation matrix. In this table k_0 is set to k_0=5 and the performance can be increased by increasing k_0. ***** add the results of Pd and Pfa *****

F. Feasibility of SNIPER

To show the feasibility of the SNIPER, we have implemented a prototype of the SNIPER for per-flow size estimation in Mininet which is a network testbed for developing OpenFlow and SDN experiments [25]. We emulates the Geant network and feed it with real traffic traces (see Table I). Table V summarizes the results of our implementation of the SNIPER framework in Mininet, demonstrating the effectiveness and feasibility of the SNIPER in production environments. Here, the optimal sampling matrix is designed using genetic algorithm for different sampling ratios.

VI. Discussion and Conclusion

Table VI compares the SNIPER with the state of the art SDN frameworks that have been used for *fine-grained* network performance measurements. It is clear that there is a tradeoff between different parameters that must be considered in the design of effective network measurement systems. Among these, SNIPER can be used in a wide range of network monitoring applications under hard network resource constraints while, it is still capable of providing estimates of IAI with acceptable accuracy and reliability, based on the application. The computational complexity and communication overhead

Framework	Application(s)	Learning Alg.	Measurement & Inference Technique	Required Resources	Measurement Accuracy	Complexity
SNIPER	Per-flow size/delay/loss /throughput monitoring	Supervised + Adaptive	Direct Measurement + Matrix completion	☺	☺	- Low computational complexity & communication overhead - No feasibility constraint
iSTAMP [3]	Per-flow size monitoring	Adaptive (Online)	Direct measurement + Compressive Sensing Inf.	☺	☺ +	- Low computational complexity & communication overhead - With Feasibility constraint
Opennetmon [4]	Per-flow size/delay/loss /throughput monitoring	NA	Fully direct	☹	☺	- High communication overhead - No feasibility constraint

TABLE VI: A comparison between SDN network measurement frameworks.

of the SNIPER frame work are low while it does not suffer from feasibility constraints as in [3].

Such capabilities are based on the fact that, in the SNIPER framework, the optimal observation matrix required by matrix completion techniques is designed using EOAs to provide the most informative measurements with the smallest amount of resources. The effectiveness of this framework has been examined using both synthetic and real network measurement traces from different network topologies and by considering two main applications: per-flow size and per-flow delay estimations. In addition, the feasibility of our framework was verified by implementing a prototype of SNIPER in Mininet.

REFERENCES

[1] L. Jose, M. Yu, and J. Rexford, “Online measurement of large traffic aggregates on commodity switches,” *ACM-Hot-ICE*, 2011.

[2] M. Yu, L. Jose, and R. Miao, “Software defined traffic measurement with opensketch,” *ACM-USENIX*, 2013.

[3] M. Malboubi, L. Wang, C. Chuah, and P. Sharma, “Intelligent sdn based traffic (de)aggregation and measurement paradigm (***i***stamp): Technical report,” *IEEE INFOCOM*, 2014.

[4] N. van Adrichen, C. Doerr, and F. Kuipers, “Opennetmon: Network monitoring in openflow software-defined networks,” *IEEE, Infocom*, 2014.

[5] Q. Zhao, Z. Ge, J. Wang, and J. Xu, “Robust traffic matrix estimation with imperfect information: Making use of multiple data sources,” *ACM-SIGMETRICS*, 2006.

[6] M. Malboubi, C. Vu, C.-N. Chuah, and P. Sharma, “Decentralizing network inference problems with multiple-description fusion estimation (mdfe),” *IEEE INFOCOM*, April, 2013.

[7] H. Nguyen and P. Thiran, “Network loss inference with second order statistics of end-to-end flows,” *ACM-IMC*, 2007.

[8] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, “Opentm: traffic matrix estimator for openflow networks,” 2010.

[9] M. Elad, “Optimized projections for compressed sensing,” *IEEE TRANS. On Signal Proc.*, vol. 55(12), pp. 5695–5702, 2007.

[10] R. M, Y. Zhang, W. Willinger, and L. Qiu, “Spatio-temporal compressive sensing and internet traffic matrices,” *IEEE/ACM Transactions on Networking*, vol. 20, pp. 662–676, 2012.

[11] G. Gursun and M. Crovella, “On traffic matrix completion in the internet,” *ACM, IMC*, 2012.

[12] Y. Liao, W. Duy, P. Geurtsz, and G. Leduc, “Decentralized prediction of end-to-end network performance classes,” *ACM, CoNEXT*, 2011.

[13] E. Candes and B. Recht, “Exact matrix completion via convex optimization,” *Foundations of Computational Mathematics*, vol. 9, p. 717772, 2009.

[14] E. Candes and Y. Plan, “Matrix completion with noise,” *Proc. of the IEEE*, vol. 98, pp. 925–936, 2010.

[15] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, “Openstate: Programming platform-independent stateful openflow applications inisde the switch,” *ACM SIGCOMM*, 2014.

[16] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, “Flow-level state transition as a new switch primitive for sdn,” *ACM SIGCOMM (HotSDN)*, 2014.

[17] R. H. Keshavan, S. Oh, and A. Montanar, “Matrix completion from a few entries,” *CoRR*, 2009.

[18] E. Talib in *Methaheuristics: From Design to Implementation*, John-Wiley, 2009.

[19] V. Kachitvichyanukul, “Comparison of three evolutionary algorithms: Ga, pso and de,” *Industrial Engineering and Management Systems*, vol. 11, pp. 215–223, 2012.

[20] R. Eberhart and Y. Shi, “Particle swarm optimization: developments, applications and resources,” *Proceedings of the 2001 Congress on Evolutionary Computation*, 2001.

[21] “Abilene traffic.” www.cs.utexas.edu/ yzhang/research/AbileneTM.

[22] “Geant network:.” http://totem.info.ucl.ac.be/dataset.html.

[23] A. Pietro, D. Ficara, S. Giordano, F. Oppedisano, and G. Procissi, “End to end inference of link level queueing delay distribution and variance,” *IEEE SPECTS 2008*, 2008.

[24] J. Ledlie, P. Gardner, and M. I. Seltzer, “Network coordinates in the wild,” *In Proc. of USENIX NSDI*, 2007.

[25] “Miniet,” At: http://mininet.org/.