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

Mehdi Malboubi[1], Yanlei Gong[2], Wang Xiong[2], Chen-Nee Chuah[1], Puneet Sharma[3]
[1] Univ. of California at Davis, USA; [2] Univ. of Electronic Science and Technology of China; [3] HP Labs, CA, USA.
{mmalboubi,chuah}@ucdavis.edu, gyl0511@gmail.com, wangxiong@uestc.edu.cn, puneet.sharma@hp.com

***Abstract*—A key requirement for network management is the accurate and reliable monitoring of relevant network characteristics. In today's large-scale networks, this is a challenging task due to the scarcity of network measurement resources and the hard constraints that this imposes. This paper proposes a new framework, SNIPER, which leverages the flexibility provided by Software-Defined Networking (SDN) to design the optimal observation or measurement matrix that can leads to the best achievable estimation accuracy using Matrix Completion (MC) techniques. To cope with the complexity of designing large-scale optimal observation matrices, we use the 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 for per-flow size and delay estimations. Our results show that SNIPER can be applied to a variety of network performance measurements under hard resource constraints. For example, by measuring only 8.8% of all per-flow path delays in Harvard network [1], congested paths can be detected with probability of 0.94. To demonstrate the feasibility of our framework, we also have implemented a prototype of SNIPER in Mininet.**

I. Introduction

The direct measurement of network's Internal Attributes of Interest (IAI) such as the per-flow size, delay, throughput or packet loss, can be challenging and infeasible due to scale, complexity and limited availability of network measurement resources. In large-scale networks, the measurement resources, including the Ternary Content Addressable Memory (TCAM) entries, processing power, storage capacity and available bandwidth, are very limited, and hence, per-flow direct measurements are infeasible. To cope with scalability issues, Network Inference (NI) techniques can be leveraged to estimate various IAI based on partial passive and/or active measurements. However, NI problems are mainly formulated as Under-Determined Linear Inverse (UDLI) problems which are naturally ill-posed in the sense that the number of measurements are not sufficient to uniquely and accurately determine the solution. Hence, side (supplementary) information from different sources and perspectives must be incorporated into the problem formulation to improve the estimation accuracy [2] [3] [4].

Software-Defined Networking (SDN) provides data plane and control plane separation enabling capability to dynamically control and re-program network switches. Most current research has focused on leveraging SDN flexibility to implement complex network management and control applications, such as enhanced route control [5] [6]. However, SDN can also enable adaptive and efficient implementation of passive and active network monitoring applications that can be controlled dynamically at run-time [7] [8] [9] [10]. This is important for many network management and security applications.

Network inference techniques can utilize the real-time programmability provided by the SDN to optimize and facilitate the process of collecting the required direct measurements and/or side information. 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 monitoring and network security applications, such as, network traffic measurement or identifying Heavy Hitters (HH) and Hierarchical Heavy Hitters (HHH). In [7], [8] and [11], 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 [12], OpenTM directly measures a traffic matrix by keeping track of statistics for each flow. Recently, in [9], 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 [10] establishes a general framework (called Opennetmon) where accurate measurements of per-flow throughput, packet loss and delay can be directly conducted.

However, these state of the art SDN-enabled network measurement and inference methods suffer from the following challenges. First, pure SDN passive and active measurement systems (e.g. [12] and [10]) can not provide per-flow measurements of all IAI, due to the hard constraint of measurement resources. Second, passive SDN-based network inference methods, such as [7], [8], [11] and [9], are not generally applicable to a variety of network performance measurement purposes; in fact, their applications are mainly limited to network traffic size estimation. Among these, the main focus in [7], [8] and [11] is on the network sub-population size estimation; in fact, they can not provide the complete visibility of the all flows over long time-horizon and they do not directly optimize the usage of available measurement resources (e.g. flow-table entries). On the other hand, [9] addresses these two later issues and it is also able to provide fine-grained estimation of all network flows using compressive sensing inference techniques with optimal *aggregated* measurements under hard resource constraints of TCAM entries. However, it might suffer from aggregation feasibility, that is, constructing required optimal aggregated measurements may not be always feasible (due to switch constraints such as longest prefix matching forwarding scheme). Also, due to the complexity of the process, it does not directly optimize the ultimate estimation accuracy in the process of providing the optimal aggregated measurements, which can lead to the degradation of performance.

Matrix Completion (MC) techniques have long 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 [13] [14] [15]. Examples of the matrix of IAI include a matrix where each entry is an Origin-Destination Flow (ODF) at different times [13], or per-flow delay/packet-loss between different nodes of the network [15]. 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 [13] [14]. Consequently, missed or non-observed entries can be estimated from a sub-set of randomly measured entries [16]. In the theory of matrix completion, the matrix of IAI can be completely reconstructed from a sub-set of randomly observed/measured entries if the number of *randomly* chosen observations is *high* enough [16] [17]. Therefore, in [13] and [14] the MC methods are used for network Traffic Matrix (TM) completion to estimate the missed entries of the TMs. Also, in [15] a new MC technique is used to predict the status of path delays or bandwidths from a set of active measurements.

In this paper, we propose a novel approach to combine SDN programmability with MC techniques where: 1) under hard constraint of measurement resources, we directly measure a sub-set of *independent* entries of the matrix of IAI without any aggregation feasibility constraints; 2) we use MC techniques to estimate unobserved entries of IAI. Accordingly, we can address the challenges in SDN-enabled network monitoring systems, in [12] [10] [7] [8] [11] and [9], where fine-grained estimation of all IAI, under resource constraint and without aggregation feasibility constraint, is not possible. To intelligently design such an efficient and scalable framework, which can be used for a variety of passive/active network monitoring applications in large-scale dynamic environments, we pose and answer the following question: *Given limited network measurement resources, how we can use the SDN capabilities to directly measure a sub-set of independent entries of the matrix of IAI and thus design the Optimal Observation Matrix (OOM) leading to the best possible estimation accuracy using matrix completion techniques?* However, the *direct* design of OOM for maximizing the performance of NI methods is prohibitive due to the complexity of the process [9] [18]. The underlying difficulty lies in the fact that formulating the inference process or algorithm as a function of the observation matrix into a closed-form and well-defined optimization problem that can be solved efficiently is extremely complicated and computationally complex.

Therefore, in this paper, we propose a new approach in designing the optimal observation matrix for network inference problems where we *directly* optimize the ultimate estimation accuracy of network monitoring applications. However, to cope with the inherent complexity of designing large-scale observation matrices, we use the Evolutionary Optimization Algorithms (EOA) that are suitable for the optimization problems where the main objective function can not be formulated as a well-defined mathematical function. In this framework, the evolutionary optimization algorithm precisely determines the optimal (i.e. the most informative) entries of the matrix of IAI that must be measured to achieve the best estimation accuracy using matrix completion techniques. We refer to our proposed framework as Software defined Network Inference with Passive/active Evolutionary-optimal pRobing (SNIPER). This framework has low computational and communication overheads that can be easily deployed on commodity OpenFlow-enabled routers/switches in a centralized or distributed manner. To improve the scalability of SNIPER, the OOM is designed using EOAs in both supervised and un-supervised settings, respectively with and without training data-sets.

Our main contribution in this paper is developing a novel framework to design the OOM, leading to the best possible estimation accuracy using matrix completion techniques. We effectively model this problem so that it can be efficiently solved using EOAs, where the 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, compared with random observation matrices that are usually used in MC techniques [16] [15] [13]. To demonstrate the effectiveness of SNIPER, its performance is evaluated using synthetic and real network measurement traces from different network topologies and for two main applications: network per-flow size and delay estimations. In addition, a prototype of SNIPER is implemented in Mininet. Due to space limitation, we occasionally refer to [19] for the detailed discussion and additional results.

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

II. SNIPER: System Description

In this section we describe various components of our proposed SNIPER framework. Figure 1 shows the general block diagram of SNIPER framework where the controller interacts with Software Defined Measurement Network (SDMN) using Network Measurement Controlling (NMC) messages to dynamically program/re-configure the SDMN and poll the required measurements and statistics. The SDMN consists of: 1) a set of Probing Agents (PA), for example hosts, which can inject probing packets into the network, and 2) a sub-set of OpenFlow Switches (OFS) in the operating network which can efficiently route the probing packets and measure the IAI. Without loss of generality, we assume that SDMN guarantees all required IAI are *observable* and *measurable*. The NMC messages include passive and active network measurement control messages that indicate which IAI must be accurately measured at different times and/or locations and setups appropriate flow-table entries and probe requests in the SDMN, accordingly. In SNIPER framework, the network measurement

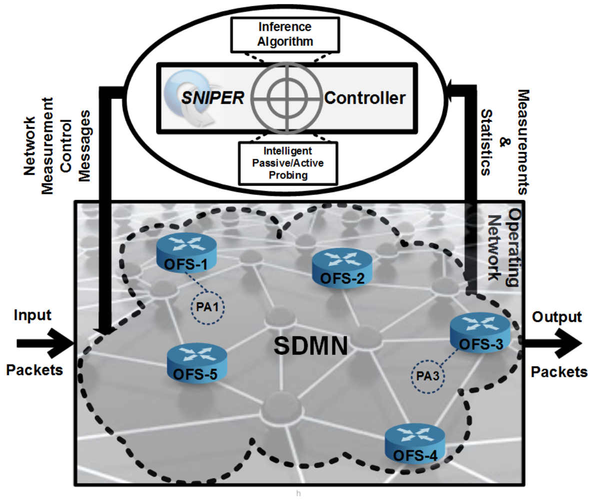

Fig. 1. SNIPER network measurement framework: a general perspective.

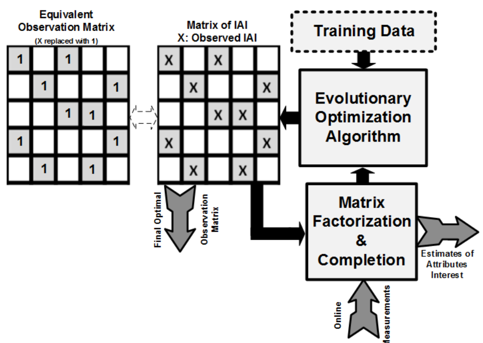

Fig. 2. Evolutionary optimal network measurement and inference process.

process consists of two stages, namely the learning and measurement epochs, as it is shown in Figure 2. In the supervised learning stage the optimal (i.e. the most informative) IAI that must be directly measured are computed using an evolutionary optimization algorithm and a training data-set. Then, in the online measurement epoch, the SDN flexibility is used to reconfigure the SDMN and to collect the measurements of corresponding optimal IAI. By decreasing and removing the dependency of SNIPER on the initial training data-set and adaptively updating the OOM, the scalability of this framework is remarkably improved for network monitoring applications in dynamic environments (please refer to Section V-D and [19]).

The controller of SNIPER can both pre-configure or adaptively reconfigure the flow-tables of the OFSs in SDMN. For passive per-flow size measurement, NMC messages (defined as passive probes, here) reconfigure the OFSs of the SDMN by installing required OpenFlow rules with appropriate forwarding actions in the flow tables for the incoming packets. The controller also requests and collects per-flow counter statistics to measure and reconstruct the matrix of per-flow sizes. On the other hand, for active network performance measurements (e.g. per-flow delay/loss/throughput), appropriate paths are first determined (for all required entries of the matrix of IAI). Then, NMC messages are used to actively probe the operating network by: 1) adaptively configuring the flow-tables of the SDMN and their forwarding actions for probing packets sent by a set of probing agents; 2) interacting with the probing packets injected into the network at the origin of the paths, and 3) collecting required measurements at destinations. Accordingly, SNIPER can compute the required IAI and obtain information of active flows and monitor the end-to-end network performance measurements. These measurements are transmitted to the controller of SNIPER where matrix completion techniques are used as the main NI algorithm to estimate all unknown IAI.

The feasibility of such SDN based measurement frameworks has been independently investigated in [9] and [10] where the capability of OpenFlow switches can be effectively utilized to measure and infer the IAI of the operating network. Likewise, SNIPER framework is capable of providing the set of partial observations of the following per-flow measurements: 1) per-flow sizes [9] by installing the flow ID prefixes in the flow tables and poll the statistics; 2) per-flow throughput [10] by determining specific path for each flow and query 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 [10] 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 [10] 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.

However, unlike [10] where all IAIs are directly measured, SNIPER directly measures a set of partial measurements and uses matrix completion techniques to infer the others. This is of particular importance for network monitoring under hard resource constraints where the amount of resources for passive per-flow size measurements (i.e. flow-table/TCAM entries) and for active per-flow performance measurements (mainly required probing bandwidth) are limited. Also, in SNIPER, since the processes of measuring individual IAI are independent, the set of partial per-flow measurements can be easily provided without aggregation feasibility comparing with [9] where optimal aggregated measurements are required. Here, to showcase the capability of SNIPER framework, the performance of SNIPER is evaluated in two main applications including per-flow size and delay estimations.

Note that, although SNIPER is an SDN based measurement framework, it can be also used in non-SDN networks where switches are not SDN enabled, indicating the compatibility of SNIPER with both current and future networking technologies. In addition, in SNIPER, since MC techniques only need partial measurements, the communication overhead between switches and controller is decreased. Moreover, the existence of efficient algorithms for implementing matrix completion and evolutionary optimization algorithms reduces the computational overhead in SNIPER [13] [15] [20].

A. SNIPER: Problem Statement

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 mentioned in the introduction, we model the NI problem as a Matrix Completion (MC) problem where the goal is to complete the matrix of IAI (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 an $n \times T_0$ matrix with rank $r << T_0$ and $T_0 < n$ where K entries of X are directly measured.

The theory of matrix completion [16] shows that a low-rank matrix X, with rank r, can be accurately 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 [15] [16] [21]. Since direct minimization of the rank of a matrix is difficult, MC problems is often formulated as a convex optimization problem as shown in 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 the others into zero, and $L(X, \hat{X}) := \sum_{i,j}(x_{ij} - \hat{x}_{ij})^2$. 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 $\hat{X} = U_{n\times r}V^T_{r\times T_0}$ and λ is the regularization coefficient that controls the extent of regularization. Here, the Frobenius norm of a matrix Z is defined as $\|Z\|_F^2 = \sum_{i,j}|z_{ij}|^2$.

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

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

The optimization problem Eq.(2) can be solved using different methods. In this paper, we adopt two different methods from recently proposed 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 [13] that uses an alternating least squares procedure to solve Eq.(2). The second one is the Decentralized Matrix Factorization algorithm [15], 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. SNIPER is a flexible framework and it allows other more advanced MC algorithms to be leveraged for the further improving of its performance.

Under hard resource constraints, it is crucial to design the OOM, which leads to the best achievable estimation accuracy using matrix completion techniques. To maximize the performance of MC algorithms with minimum number of required measurements, the process of designing OOM 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 the observation matrix in our case is a binary matrix, it is computationally expensive and intractable to use integer optimization techniques in such a design process for large-scale networks [9]. Therefore, in this paper, we use evolutionary optimization algorithms to cope with the inherent complexity of design process.

III. Optimal Observation Matrix Design

Evolutionary algorithms are heuristic optimization methods [22] for solving NP-hard optimization problems where the main objective function may not be formulated as a well-defined mathematical function. The evolutionary algorithms, generally, consist of three main processes. The first process is the initialization process where the initial population of individuals (i.e. representations of solution) is randomly generated. In the second process, the fitness value of each individual is evaluated and a sub-set of the best individuals is selected. In the third process, a new population is generated by the perturbation of current solutions. The algorithm continues until a stopping criterion is met. Here, we use two EOAs, namely, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Among these, the PSO has a computational advantage over GA as the PSO algorithm does not require sorting of fitness values [22] [20].

Here, to effectively apply the GA, a chromosome (representing a solution) is defined as a binary observation matrix C with size $n \times T_0$ 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, see Section. IV). 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 represented in 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 $\{1, ..., n\}$ and operator : is the colon operator in MATLAB. 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 $p_m = 0.01$. These parameter values were determined using a trial and error method [19].

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

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 $n \times T_0$ binary matrix, representing the measurement/observation matrix, and V_i^k is also an $n \times T_0$ 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 are 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 set 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. $(\sim, I_{max}^V) = sort(abs(V_i^k(:)))$ where : denotes the colon operator in MATLAB), 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. In [19] by considering a smaller network, we have shown that our evolutionary OOM design procedures can converge to the global optimum which indicates the effectiveness of our method in designing the OOM.

IV. PERFORMANCE EVALUATION METHODOLOGY

The performance of SNIPER is evaluated in two main applications, namely per-flow size and delay estimations. For this purpose three network topologies, including Abilene, Geant and Harvard networks, and both synthetic and real network traces are considered. For per-flow size estimation, we use real traffic traces from Abilene [23] and GEANT [24] networks; the characteristics of these traffic traces are presented 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 [25], 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 $0 \le \lambda \le 10$ which includes both low and high noise scenarios. In addition, we use real per-flow delay from Harvard study [1] which contains 2,492,546 measurements of application-level RTTs, with timestamps, between 226 Azureus clients collected in 4 hours [15].

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

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 $\mathcal{T}_0$ is the duration of the traffic in Table I and $T_0 = \left\lceil \frac{\mathcal{T}_0}{t_p} \right\rceil$) is utilized to design the OOM using the GA and PSO evolutionary algorithms. The matrix with the best fitness value in the last population of the learning stage determines the OOM and its estimation performance is denoted by subscript T_0 in our results. Then, the same OOM is used over other $t_p - 1$ parts of the data-set (called measurement epochs) and the average of the performance over multiple parts is computed and is denoted by subscript Avg in our results. In the un-supervised learning scheme, the procedure is the same; however, the OOM is designed by minimizing the norm of the observation matrix using GA and without training data-set (see Section V-D). The number of measurement paths, denoted by K, plays an important role in improving the estimation accuracy. This parameter is defined as $K = s \cdot (n \cdot T_0)$ where s is the Sampling Ratio (SR) and $0 \le s \le 1$. Thus, given sampling ratio s and having n and T_0, consequently, K (i.e. the number of required measurements) and the amount of required resources can be computed. Here, low values of the sampling ration s (i.e. smaller Ks) indicates the hard-resource constraint regime. 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 subscripts are used to distinguish between different applications where CP denotes Congested Paths and HH denotes Heavy Hitters, respectively.

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

Network	Date	Duration	Resolution	TM Size ($n \times T_0$)
Abilene [23]	2004-05-01	1 week	5 min.	144 × 2016
GEANT [24]	2005-01-08	1 week	15 min.	529 × 672

TABLE I. REAL DATASETS UNDER STUDY.

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

V. THE APPLICATIONS OF SNIPER FRAMEWORK

In this section, we showcase the effectiveness of SNIPER for two main applications, namely per-flow delay and per-flow 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_0 is set to $T_0 = 100$; 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 OOM 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. Per-Flow Delay Estimation using SNIPER

Figures 3 shows the performance of SNIPER in the estimation of per-flow delay on Abilene network using synthesis data generated using the model in Eq.(5) where the MC technique is DMFSGD as in [15]. Here, the OOM is designed using the GA and only by considering the propagation delay in Eq.(5) in the learning epoch. Then, this OOM 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 according to Eq.(5) that models the network paths delay [25]. This figure shows that at low sampling ratios (SR), indicating the hard resource constraint regime, the optimal observation matrix designed by the GA can obtain a better estimation accuracy, with more robust performance against noise. To see the same performance behavior on Geant network please refer to [19]. Figure 4 also shows the performance of SNIPER framework on real per-flow delay estimation in Harvard network [1]. This figure also shows that at low sampling ratios (SR), indicating the hard resource constraint regime, the optimal observation matrix designed by the GA can obtain a better estimation accuracy. In addition, Table II indicates the capability of SNIPER framework in the reliable detection of congested paths with high probability of detection P_{CP}^d and low probability of false alarm P_{CP}^{fa}. It is clear that, by increasing sampling ratio the accuracy of estimation is improved. However, at lower sampling ratios and comparing with random sampling strategy in [15], SNIPER can obtain a better estimation accuracy and more reliable detection performance, due to the intelligent design of the OOM. For example, by measuring 8.8% of per-flow path delays, congested paths can be detected with probability 0.94 in Harvard network.

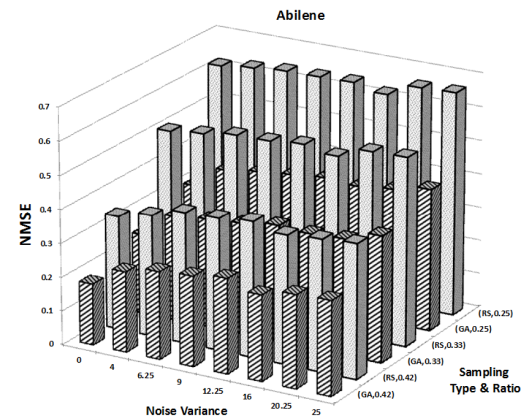

Fig. 3. The $NMSE$ vs. SR & noise for Abilene network.

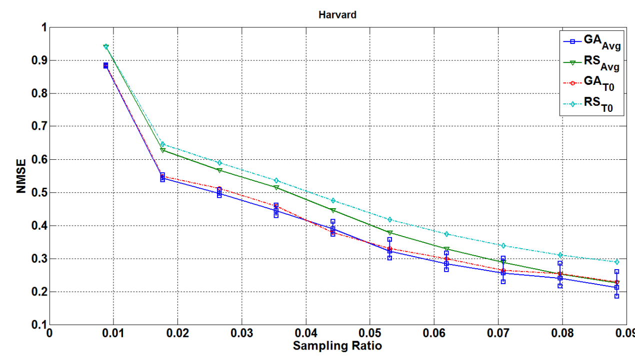

Fig. 4. The $NMSE$ for Harvard network in different sampling ratios.

This is an important factor in active network performance measurement where the network monitoring bandwidth is very limited.

It should be noted that, in Figure 4, and throughout this paper, the blue squares represent the minimum and maximum of $NMSE$ (or $NMAE$) for each sampling ratio using our EOA based sampling strategy. Also, in low sampling ratios and in all measurement epochs a better estimation accuracy is obtained using SNIPER framework.

B. Per-Flow Size Estimation using SNIPER

Figure 5 shows the performance of 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-base as in [13]. Again, it is clear that, comparing with random sampling strategy originally proposed in [13], the performance of the matrix completion in SNIPER framework is improved by the design of the optimal (i.e. the most informative) observation matrix. Note that, although the estimation accuracy is improved by increasing the sampling ratio, the performance of SNIPER is better at low sampling ratios,

SR	0.0177	0.0354	0.0531	0.0708	0.0885
P_{CP}^d	0.6080	0.7534	0.8358	0.9061	0.9377
P_{CP}^{fa}	0.1620	0.1262	0.0750	0.0530	0.0398

TABLE II. AVERAGE P_{CP}^d AND P_{CP}^{fa} FOR HARVARD NETWORK.

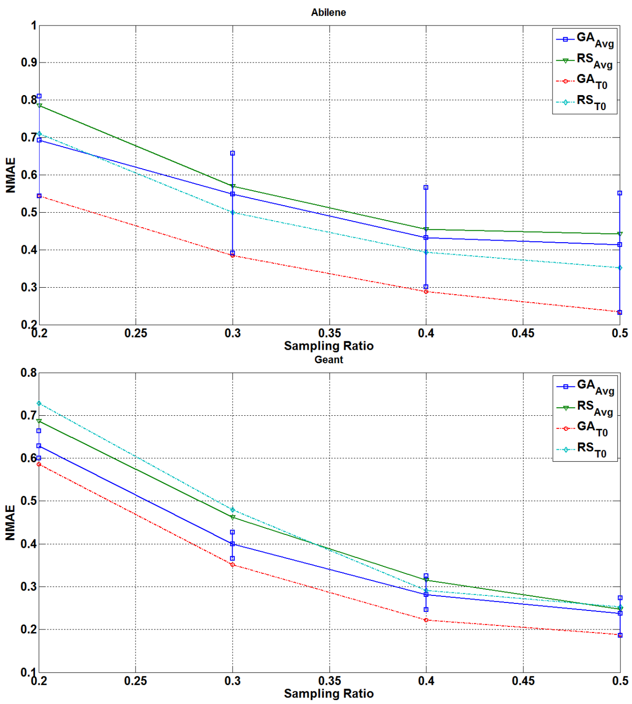

Fig. 5. $NMAE$ vs. sampling ratio for Abilene and Geant networks.

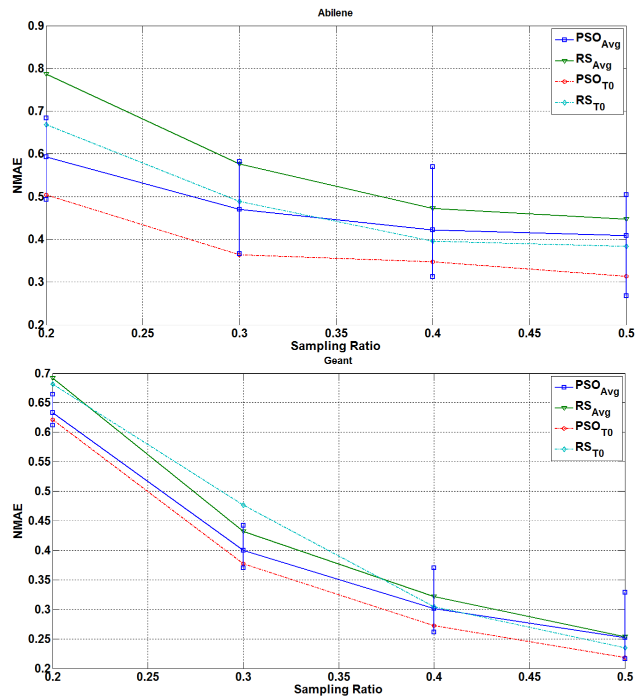

Fig. 6. $NMAE$ vs. sampling ratio for Abilene and Geant networks.

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

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

which indicates the case of hard resource constraint of TCAM entries as the main resource for per-flow size measurement. Also, a better estimation accuracy is obtained almost for all sampling ratios and in all measurement epochs. Table III also shows the average performance of SNIPER framework in the reliable detection of heavy hitters under low sampling ratios. For example, by measuring 20% of all the per-flow sizes, HHs can be detected with probability 0.65 and 0.78 in Abilene and Geant networks, respectively. This has a great implication in applications, such as, network traffic engineering and security.

C. Scalability of SNIPER

As it was shown, SNIPER can improve the estimation accuracy under hard resource constraint regimes. To reduce the high computational complexity of the GA in designing the OOM in large-scale networks and increase the scalability of SNIPER framework, here, we use the PSO evolutionary optimization algorithm which is much faster than the GA [22] [20] and it can reduce the computational complexity and processing power of SNIPER. Figure 6 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 time using the PSO algorithm is quantified using the notion of Processing Gain 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. The processing gains for Abilene and Geant networks are PG=56% and PG=65%, respectively.

D. Unsupervised SNIPER

In the case of supervised learning, 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). In the lack of training data-set, it is important to decrease or remove the dependency of SNIPER on the initial training data-set.

During the course of this study, we observed that there is a strong correlation between the ultimate performance of the MC technique, using the best observation matrix designed by the GA in each iteration (denoted by S_Ω), and the norm of that observation matrix (i.e. the maximum singular value of S_Ω denoted by $\sigma_1(S_\Omega)$). In fact, we observed a correlative decreasing behavior between the ultimate estimation accuracy (represented by $NMAE$) and $\sigma_1(S_\Omega)$. To quantify this correlation for each sampling rate, we compute the Correlation Coefficient (denoted by ρ) between $NMAE$ and the norm of observation matrix where for any two vectors a and b, ρ is defined as $\rho = \frac{Cov(a,b)}{\sqrt{Var(a)Var(b)}}$. Accordingly, the average correlation coefficient over different sampling rates for both Abilene and Geant networks are ρ_{Avg} =0.89 and ρ_{Avg} =0.92, respectively.

Net./Prm.	SR	GA_{Avg}	GA^{σ}_{Avg}	$PG\%$
Abilene	0.3	0.5561	0.5464	93 %
GEANT	0.2	0.6014	0.6646	88 %

TABLE IV. $NMAE$ FOR FITNESS FUNCTIONS $NMAE$ AND $\sigma_1(S_\Omega)$ (DENOTED BY σ).

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. $NMAE$ FOR GEANT NETWORK IN MININET.

Therefore here, instead of minimizing the ultimate estimation accuracy, the norm of the observation matrix (i.e. $\sigma_1(S_\Omega)$) is considered as the fitness function and it is minimized using our GA. In this way, we can cope with the complexity of the formulating and designing of binary observation matrices with minimum norm; moreover, the dependency of SNIPER framework on the training data set is removed. Table IV shows the performance of this method in TM completion indicating two important facts which are of crucial importance for MC in large-scale networks and they can remarkably improve the scalability of SNIPER framework. First, in the absence of training data, minimizing $\sigma_1(S_\Omega)$ is almost as effective as directly minimizing $NMAE$ using training data-set. Second, the computational complexity of the GA with fitness function $\sigma_1(S_\Omega)$ is much less than directly minimizing the $NMAE$ where $PG\% := 100 \times \frac{PT_{NMAE}-PT_\sigma}{PT_{NMAE}}$, and PT_{NMAE} and PT_σ respectively denote the processing times for running GA with fitness functions $NMAE$ and $\sigma_1(S_\Omega)$. Note that, the processing gain can be further improved by designing the OOM using PSO with fitness function $\sigma_1(S_\Omega)$.

In [19] we also demonstrate how the OOM designed in the learning stage can be adaptively updated based on the most current behavior of the network. This is of particular importance for applying SNIPER framework in dynamic environments.

E. Feasibility of SNIPER

To show the feasibility of SNIPER, we have implemented a prototype of SNIPER for per-flow size estimation in Mininet [26]. We implement the controller of SNIPER in POX and, also, we emulate the Geant network and feed it with real traffic traces (see Table I). Table V shows the implementation results of SNIPER framework in Mininet (using OOM designed by GA), demonstrating the feasibility of the implementation of SNIPER framework in production environments.

VI. CONCLUSION

In this paper, we introduced SNIPER, an intelligent network measurement framework, where the flexibility provided by SDN and the capability of EOAs are used to optimally design the observation matrix which leads to the best possible estimation accuracy via applying matrix completion techniques. We showed that, under hard resource constraints, SNIPER is an efficient and scalable framework that can be used in a wide range of network monitoring applications in large-scale networks and without aggregation feasibility constraints.

VII. ACKNOWLEDGMENT

This work supported by National Science Foundation CNS-1321115 grant and HP Labs Innovation Research Award.

REFERENCES

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

[2] 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.

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

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

[5] A. Gupta and et. al., “Sdx: A software defined internet exchange,” *ACM SIGCOMM (HotSDN)*, 2014.

[6] C. E. Rothenberg and et al, “Revisiting routing control platforms with the eyes and muscles of software-defined networking,” *ACM SIGCOMM (HotSDN)*, 2012.

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

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

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

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

[11] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, “Dream: Dynamic resource allocation for software-defined measurement,” *ACM SIGCOMM (HotSDN)*, 2014.

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

[13] 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.

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

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

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

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

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

[19] M. Malboubi, Y. Gong, W. Xiong, C. Chuah, and P. Sharma, “Software defined network inference with passive/active evolutionary-optimal probing (sniper): Technical report,” tech. rep., 2015. At: http://web.ece.ucdavis.edu/cerl/techreports/2015-2/.

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

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

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

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

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

[25] 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.

[26] “Mininet,” At: http://mininet.org/.