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: University of California at Davis, {mmalboubi,chuah@ucdavis.edu}
2: University of Electronic Science and Technology of China, {wangxiong@uestc.edu.cn}
3: Hewlett-Packard Laboratories, {puneet.sharma@hp.com}

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 feasibility of our framework, we have also implemented a prototype of SNIPER in Mininet environment.

1. INTRODUCTION

In computer networks, network management refers to the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance, provisioning and security of networked systems [1]. 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 directly measured or indirectly inferred. In addition, accurate network monitoring is essential for network management in order to reach QoS agreements.

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 including the limited number of Ternary Content Addressable Memory (TCAM) entries at switches, the limited processing power and storage capacity and limited available bandwidth in active network performance measurement.

Network Inference (NI) techniques are powerful network monitoring tools that can help estimate the IAI based on a limited set of measurements and, accordingly, mitigate the limitations and constraints of direct network measurement techniques. NI techniques are usually formulated as under-determined inverse problems which are naturally ill-posed problems, that is, 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 [2] [3] [4].

Recently, Matrix Completion (MC) techniques have been used as network inference tools where the problem is the completion of a matrix of IAI from the direct measurement of a sub-set of its entries [5][6][7]. The main assumption in MC techniques is that the matrix of IAI is a low-rank matrix which contains redundancies and thus not all of its entries are needed to represent it. In the theory of matrix completion, the matrix of IAI can be completely reconstructed from a sub-set of observed entries (i.e. measurements) if the number of *randomly* chosen observations are *high* enough [8][9]. Accordingly, in [5] and [6] the MC methods are used for network Traffic Matrix (TM) completion to estimate the missed entries of the TMs. Also, in [7], a new MC technique has been used for active network performance measurements where the status of path delays or available bandwidths are predicted from a set of active measurements and using a new MC technique. Since in communication networks a variety of resources of the communication infrastructure at different layers are shared, the MC methods rely on the fact that the matrices of IAI in network monitoring applications contain spatio-temporal redundancies (i.e. correlations or

structures) that can be used to estimate missed or non-observed entries from a sub-set of randomly measured entries.

Software-Defined Networking (SDN), along with its OpenFlow enabler, is an emerging technology that nicely separates the measurement data plane and control plane functions, and provides a capability to control/re-program the internal configurations of switches in dynamic environments. Consequently, SDN allows to adaptively and efficiently implement more complex network monitoring applications, including both passive and active network measurements, without the need of customization [7] [10] [11] [12]. For passive network measurement, the most SDN based works related to traffic engineering and network security applications where the main goal is focused on network traffic measurement or identifying aspects of the network traffic such the presence of Heavy Hitters (HH) and Hierarchical Heavy Hitters (HHH). In [11], the authors propose a reconfigurable measurement architecture for hierarchical heavy hitter detection, and then in [12], they propose a re-programmable structure (called OpenSketch) where a variety of sketches for different measurement tasks can be defined and installed by the operator. In [13], OpenTM estimates a traffic matrix by keeping track of statistics for each flow. Recently, in [10], we have proposed an intelligent SDN based traffic measurement framework (called iSTAMP) with the ability of adaptive and accurate fine-grained flow estimation. In addition, for active network measurement under SDN paradigm, the very recent work [14] establishes a general framework where accurate measurements of flow throughput, packet loss and delay can be obtained.

Accordingly, in network monitoring applications, the flexibility provided by the SDN can be used to intelligently and adaptively measure the entries of the matrix of IAI. Hence, an interesting question that can be asked is as follows:

Under hard resource constraints of network measurement resources, how can we optimally measure or sample the entries of the matrix of IAI and design the optimal observation matrix which leads to the best possible estimation accuracy via using matrix completion techniques?

However, the *direct* design of observation matrices for maximizing the performance of NI methods is prohibitive due to the complexity of the process [10][15]. To simplify the process of designing the optimal observation matrix other objective functions (e.g. coherency [15] or condition number [3]) are considered in the optimization process, while accepting the unavoidable sacrifice in performance [15]. The underlying difficulty in this direct optimal observation matrix design is that formulating the network inference process or algorithm into a closed-form and well-defined mathematical objective function that can be efficiently optimized is extremely complicated and computationally complex, if it is not impossible or intractable.

Therefore, in this paper, we change the main 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, under the very hard constraints of measurement resources in network monitoring applications, the evolutionary optimization algorithm acts as a SNIPER which precisely captures or measures the best entries of the matrix of IAI which leads to best estimation accuracy via matrix completion techniques.

Under hard resource constraints of network measurement resources, SNIPER is simple, generic, and efficient with the ability to optimally measure the most informative IAI which lead to the best achievable estimation accuracy via matrix completion methods. In fact, SNIPER 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. Here, we build upon the recent achievements in the theory of matrix completion to develop a practical foundation for guiding the design of optimal measurement or observation matrices under hard resource constraints of network measurement resources.

Our main contributions are summarized as follow:

- To the best of our knowledge for the first time, we use the EOAs to design the optimal observation matrix where ultimate network inference performance is the main objective function to be optimized.
- We evaluate the performance of SNIPER using both synthetic and real network measurement traces from different network topologies by considering two main applications including network traffic and delay estimation. Furthermore, we implement a prototype of SNIPER and demonstrate its feasibility and effectiveness in Mininet environment.

The rest of this paper is organized as follows. Section 2 provides an overview of SNIPER and the matrix completion that we have used as our main NI technique. In Section 3 we describes our optimal observation matrix design procedure using the EA algorithms. Then, in Section 4, we explain our methodology for evaluating the performance of the SNIPER. Accordingly, in

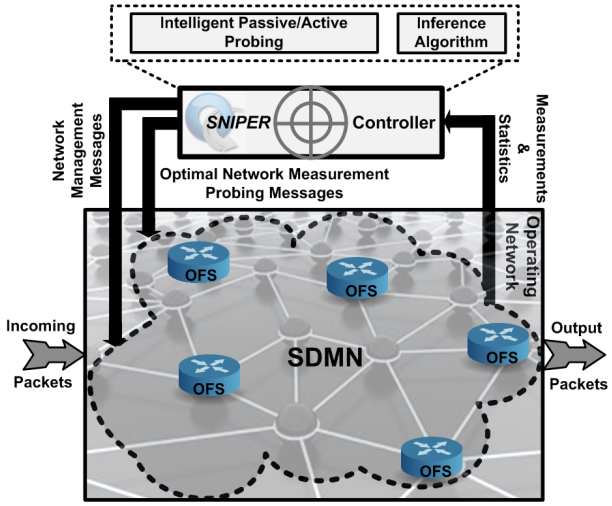

Figure 1: SNIPER network measurement framework: a general perspective.

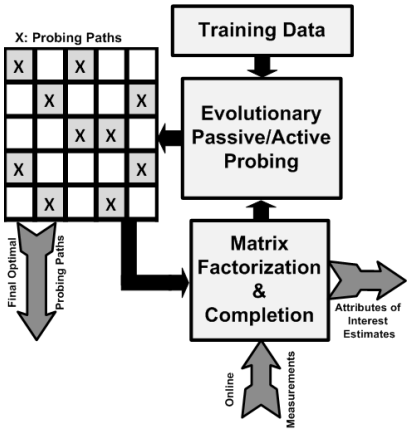

Figure 2: Evolutionary optimal network probing process.

Section 4 we evaluate the performance of SNIPER considering two main applications including per-flow path delay and flow size estimation. Section 6 summarizes our most important results.

2. 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[1]. The NM messages include regular network operating and control commands while ONMP messages include passive/active probing messages which exactly indicate which IAI must be accurately measured at different times or spaces.

In the SNIPER framework, the network measurement process is consisted of two stages, including learning and measurement stages, 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 measurement stage, the flexibility provided by the SDN is used to reconfigure the SDMN and to collect the statistics/measurements of the corresponding ONMPs. These measurements are transmitted to the SNIPER controller where matrix completion techniques are used as the main NI algorithm to estimate all unknown attributes of interest. The SNIPER controller can both preconfigure the switch with forwarding rules as well as it can reactively respond to requests from switches, which are sent when a packet matches none of the existing rules enters the network. Besides managing the forwarding plane, the OpenFlow based SNIPER controller is also capable of requesting per-flow counter statistics and injecting packets into the network. Accordingly, the SNIPER architecture can obtain information of running flows and regularly injects packets into the network to monitor end-to-end network performance measurements [10][14].

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. However, 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 pre-defined network measurement packets (probes) which will be captured by reconfigured OFSs and provide corresponding statistics.

In [14], and we, in [10], have shown that how the SDN capabilities can be effectively utilized to measure the four most important IAI of the operating network including per-flow size, throughput, and packet loss and delay. Accordingly, the SNIPER controller is capable of providing: 1) per-flow sizes [10] by installing the flow ID prefixes in the flow tables and polls the statistics; 2) per-flow throughput [14] by determining specific path for each flow and queries switches to retrieve per-flow statistics where for each query determines the amount of bytes sent and the duration of each flow; 3) per-flow packet loss [14] by polling flow statistics from the first and last switch of each path and subtracts the increase of the source switch packet counter with the increase of the packet counter of the destination switch, and 4) per-

[1]For optimal network monitor placement problem, please refer to [16] and [17] or our recent work in [18].

flow delay [14] by assigning a specific path to each flow and regularly injecting packets into the first switch and polling the statistics from the last switch and computing the difference between the packets departure and arrival times, subtracting with the estimated latency from the switch-to-controller delays. In OpenFlow, these tasks can be accomplished using *PacketIn*, *PacketOut*, *FlowMod*, *FlowRemoved*, *StatsRequest* commands (please refer to [14] and [19] for further details).

2.1 SNIPER: Problem Statement

The 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 this network. In the SNIPER framework, the NI problem is modeled as a matrix completion 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 m entries of X is directly measured; the quantity m is also defined as $m = s.(n.\mathcal{T})$ where s is the sampling ratio and $0 \leqslant s \leqslant 1$.

The theory of matrix completion [8] shows that under some suitable conditions, with high probability, X can be exactly recovered from just $O(n^b r log(n))$ *randomly* observed entries, where b is only slightly larger than 1. In practice, X is often full rank but with a rank r dominant component, that is, X has only r significant singular values (i.e. $\sigma_1 \leqslant ... \leqslant \sigma_r$) 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 [7][8][9][20]. 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, $\bar{\Omega}$ is the complement of Ω (which identifies missed or unobserved entries), P_Ω is a sampling function that preserves entries of X in Ω (i.e. that is $[P_\Omega(X)]_{ij} = x_{ij}$) and turns out the others into zero, and $L(X, \hat{X})$ is defined as $L(X, \hat{X}) := \sum_{i,j=1}^{n} (x_{ij} - \hat{x}_{ij})^2$. In fact, the MC searches for a low-rank matrix $\hat{X}$ that approximates X with sufficient accuracy at the observed entries in Ω. The missing entries in X 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 $rank(\hat{X}) \leqslant r$) is factorized as $\hat{X} = U_{n\times r} V^T_{r\times n}$ and λ is the regularization coefficient that controls the extent of regularization. Here, the Frobenius norm of a matrix Z is defined as $\|Z\|_F^2 = \sum_{i,j=1}^{n} z_{ij}^2$. The general optimization problem Eq.(2) can be solved using different methods.

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

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

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 [5] that uses an alternating least squares procedure to solve Eq.(2). The second one is the Decentralized Matrix Factorization algorithm [7],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 the attributes of interest in network monitoring applications contain temporal and/or spatial redundancies that can be used to estimate non-observed or missed entries.

Therefore, under hard resource constraints of network measurement resources, it is crucial to design the optimal observation or measurement matrix, which leads to the best achievable estimation accuracy via applying matrix completion technique onto the NI problem. To show the importance of such a design, consider the three spatial independent process $x_1(t) = \frac{1}{2}(x_1(t-1) + x_1(t+1))$, $x_2(t) = 2x_2(t-1) + 3$ and $x_3(t) = \frac{1}{2}x_3(t+1) - 10$. Using the temporal structure of these three 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 which is necessary for applying matrix completion techniques. It is clear that such an optimal observation matrix can not guaranteed to be obtained using a random sampling strategy.

To maximize the performance of NI algorithms with minimum number of required measurements, such a design process must directly target the ultimate estimation accuracy in the network monitoring application. However, it is extremely complicated and computationally complex, if it is not impossible or intractable, to formulate the ultimate performance criteria using a closed-form and well-defined mathematical objective function and solve the problem using regular integer optimization techniques. 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.

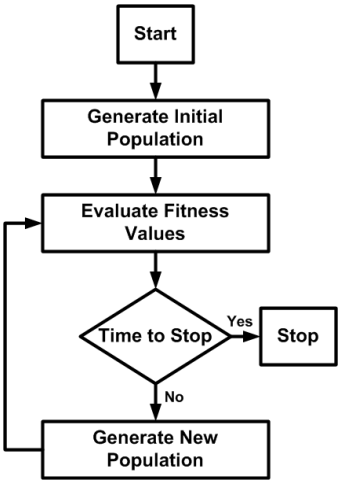

Figure 3: Evolutionary optimization algorithm's flowchart

3. EVOLUTIONARY-OPTIMAL OBSERVATION MATRIX DESIGN

Evolutionary algorithms are the sub-category of heuristic optimization methods [21] for solving NP-hard optimization problems where the main objective function may not be formulated as a well-defined mathematical function. As Figure 3 [22] 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, directly or indirectly. If an indirect representation is used, each individual must be decoded into 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 perturbation of solutions in the existing population. The algorithm is run until one or more of the stopping criteria are met. In this paper we use two EAs namely as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) which have been applied to many optimization and machine learning problems [21][22].

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 [22].

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 whereby 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 [22] [23].

Throughout this paper, the fitness of the solutions in our evolutionary algorithms (chromosome in GA and particle in PSO) are evaluated using the following two metrics in Eq.(6), namely, Normalized Mean Absolute Error (NMAE) and Normalized Mean Square Error (NMSE) where x_{ij} denotes the ij^{th} entry of the matrix X (which is known in the learning stage indicated by T_0) and $\hat{x}_{ij}$ denotes the ij^{th} entry of the matrix $\hat{X}$ which is output of the MC process.

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. 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 (as defined in Eq.(3)), with probability p_c, is applied on each pair of

parents to generate new children (offsprings); offsprings form 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.

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

Likewise, the PSO is started by generating N_p particles where i^{th} particle is identified by its position P_i^k and 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 c_1 and c_2 are acceleration constants, which here they setup to $c_1 = c_2 = 2$, 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 the new measurement paths) using two methods: 1) set the entries $\{p_{ij}^k\}_{ij \in I_{max}^V} = 1$ where I_{max}^V indicates the set of ij^{th} entries with highest velocities in the matrix V_i^k (i.e. $(\sim, I_{max}^V = sort(V_i^k(:)))$, and 2) $\{p_{ij}\}$ 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.

In both GA and PSO evolutionary algorithms, some simple manipulations are applied at each step to keep the number of measurement paths constant for each sampling rate in such a way that chromosomes/particles remain symmetric (if it is required, for example, in the case of delay measurement) with having at least an one in each row and column of the solution representation (please refer to [19] for further details).

$$V_i^k = \omega V_i^{k-1} + c_1\alpha_1(BP_i^k - P_i^{k-1}) + c_2\alpha_2(GP^k - P_i^{k-1}) \tag{4}$$

4. SNIPER PERFORMANCE EVALUATION: METHODOLOGY

Network	Date	Duration	Resolution	TM Size ($n \times T_c$)
Abilene [26]	2004-05-01	1 week	5 min.	144 × 2016
GEANT [27]	2005-01-08	1 week	15 min.	529 × 672

Table 1: Real Datasets under study.

The performance of SNIPER is evaluated onto two main applications, including per-flow path delay and per-flow size estimation 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 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 [24], 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 parameter λ is chosen such that $0 \leqslant q_{ij} \leqslant 10$. In addition, we use real per-flow delay from Harvard [25] which contains 2,492,546 dynamic measurements of application-level RTTs, with timestamps, between 226 Azureus clients collected in 4 hours [7]. For per-flow size estimation, we use publicly available traffic traces from Abilene [26] and GEANT [27] networks; the characteristics of these traffic traces are represented in Table 1.

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

In our supervised learning scheme, each data set is divided into t_p parts. The first part with size $n \times \frac{\mathcal{T}}{t_p}$ is utilized to design the ONMP messages using the GA and PSO evolutionary algorithms. The last population determines the ultimate ONMP and the it estimation performance is denoted by lower scripts T_0 in our results. Then, the same ONMP set is used over other $t_p - 1$ parts of the data set and the average of the performance over multiple parts is computed and is denoted by lower scripts Avg in our results. The number of measurement paths m plays an important role in improving the estimation accuracy. This parameter is defined as $m = s.(n.\mathcal{T})$ where s is the Sampling Ratio and $0 \leqslant s \leqslant 1$; in fact, the higher the m 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). The states of the IAI are also classified into two different classes. In the case of classifying per-flow delay estimation, 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 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 CS denotes Congested Paths and HH denotes Heavy Hitters, respectively.

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

$$P^d = \frac{1}{|\bar{\Omega}|} \sum_{ij\in\bar{\Omega}} Pr(\hat{x}_{ij} \geqslant \theta | x_{ij} \geqslant \theta)$$
$$P^{fa} = \frac{1}{|\bar{\Omega}|} \sum_{ij\in\bar{\Omega}} Pr(\hat{x}_{ij} \geqslant \theta | x_{ij} < \theta) \quad (7)$$

5. SNIPER PERFORMANCE EVALUATION: APPLICATIONS

In this section, the effectiveness of our network measurement and inference framework in Fig. 1 is justified for two main applications, including per-flow delay and size estimation, and under different configurations. Each configuration defines how the measurement process is setup in the SNIPER framework and how it evaluates the performance of the estimation. Accordingly, each configuration is determined by determining the network under study, the matrix completion technique, the length of learning period T_0, the Sampling Ratio (SR) s, the number of measurements m (where $m = s(nT_0)$) and the number of parts in the data set t_p. The type of the sampling strategy that is used to identify the ONMP in the SNIPER framework is another parameter that is used in the definition of each configuration. The sampling strategy is identified by notions RS, GA and PSO which respectively denote to the ONMP designed by random sampling and evolutionary algorithms GA or PSO. Note that the performance of follwoing results can be improved by increasing the number generations in the evolutionary algorithms. **** Add the parameters of GA and PSO ****

5.1 Per-Flow Delay Estimation using SNIPER

Figure 4 shows the performance of the SNIPER in the estimation of per-flow delay on both Abilene and Geant networks using synthesis data generated using the model in Eq.(5) where the MC technique is DMF-SGD as in [7]. Here, the ONMP is designed using the GA and only by considering the propagation delay in

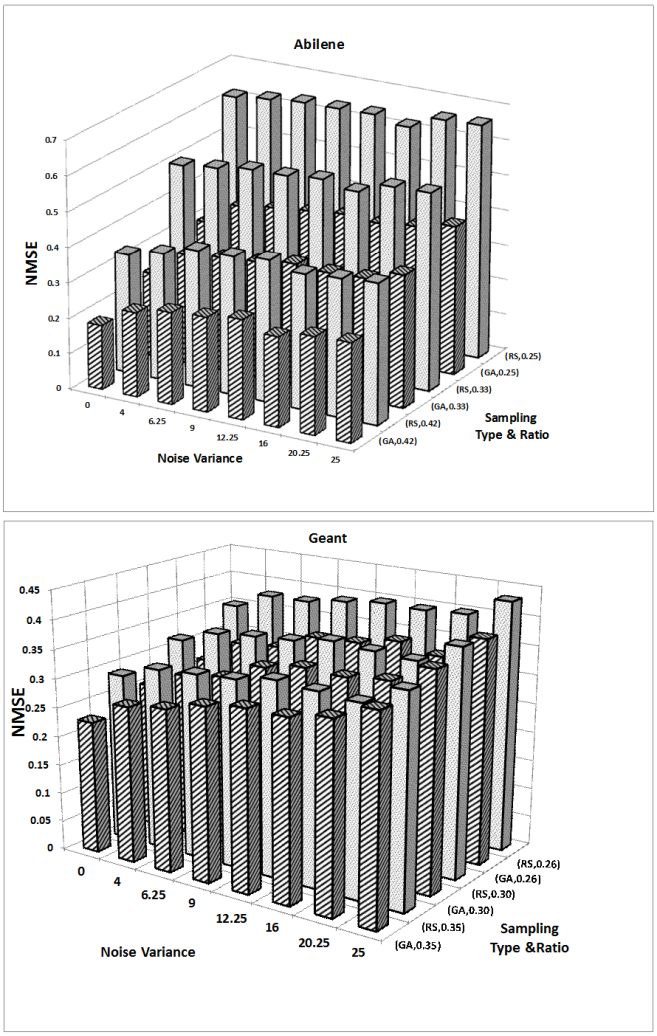

Figure 4: The average $NMSE$ for different sampling types and ratios, and variances of queuing delay.

Eq.(5) in the learning stage. Then, this ONMP is used to evaluate the performance of the MC technique on the other parts of the data set, based on the $NMSE_{Avg}$, where queuing delay is added to the propagation delay as Eq.(5) models the network paths delay [24]. Comparing with the random sampling, these figures show that for lower SRs, the ONMP designed by the GA provides better estimation accuracy. Also it is more robust against the variance of the queuing delay. This is an important factor for active network measurements under hard resource constraints. Figure 5 also shows the performance of the SNIPER framework on real per-flow delay from Harvard [25], indicating the fact that by the intelligent design of the ONMP the better estimation accuracy can be obtained via applying MC techniques. **** add the pd and pfa here for real harvard data in table ***.

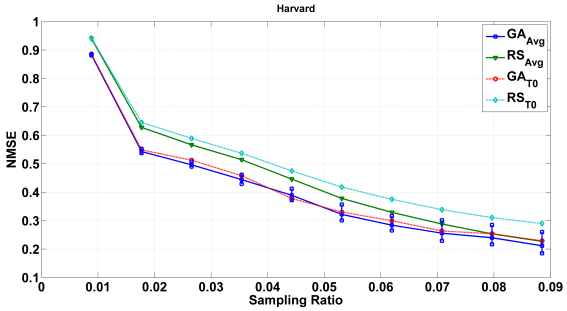

Figure 5: The average of P_{CP}^{d} and P_{CP}^{fa} for different delay variances, sampling ratios and sampling types.

	$SR = 0.2$	$SR = 0.3$	$SR = 0.4$	$SR = 0.5$
P_{HH}^{d} Abilene (RS)	0.6255	0.7544	0.8426	0.8901
P_{HH}^{d} Abilene (GA)	0.6550	0.7693	0.8380	0.8851
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
	$SR = 0.2$	$SR = 0.3$	$SR = 0.4$	$SR = 0.5$
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	0.0042	0.0038
P_{HH}^{fa} Geant (GA)	0.0095	0.0058	0.0041	0.0034

Table 2: 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.

5.2 Per-Flow Size Estimation using SNIPER

Figure 6 shows the performance of the SNIPER in the estimation of network per-flow sizes on both Abilene and Geant networks using real traffic traces (Table 1) where the MC technique is the SRSVD as in [5]. 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 under hard resource constraint of measurement resources at low sampling ratios. The better accuracy is obtained almost for all sampling ratios and in all time intervals indicated by the blue squares representing the minimum and maximum of NMAE for each sampling ratio. Table 2 also shows the average performance of the SNIPER framework for the detection of heavy hitters. In fact, comparing with random sampling strategy, this table indicates the good capability of this framework in detection HHs even under low sampling ratios.

5.3 Scalability of SNIPER

As we have seen in the previous results, the SNIPER can improve the estimation accuracy under hard constraints of measurement resources. However, since the genetic algorithm targets the ultimate matrix completion inference performance, the processing power is problematic in large-scale networks and this limits the scalability of the SNIPER framework. To reduce the computational complexity and processing power of the SNIPER we use use the PSO evolutionary optimization algorithm which is much faster than the GA [21][22] and it can reduce the computational complexity and processing power of the SNIPER. Figure 7 shows the performance of SNIPER, for per-flow size estimation in this case, representing the fact that for low sampling rates the intelligent design of the observation matrix results in a better estimation accuracy.

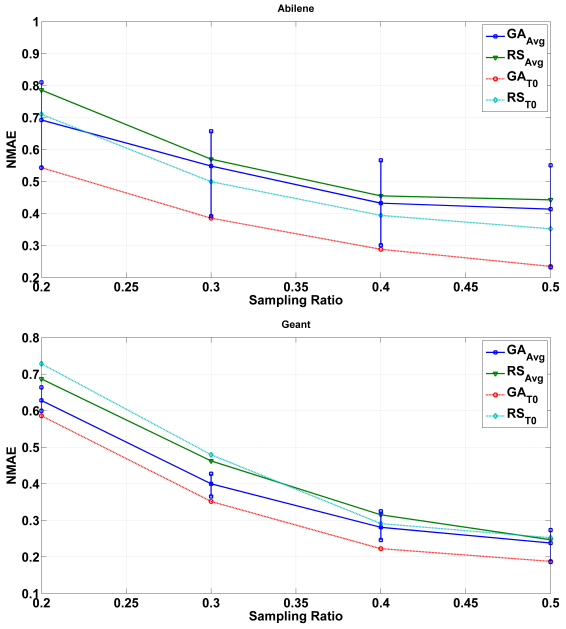

Figure 6: $NMAE$ v.s. sampling ratios.

In addition, instead of minimizing the ultimate estimation accuracy (e.g. represented by $NMAE$), the norm of the observation matrix Ω is minimized where the norm of a matrix is defined as the maximum singular value of the matrix and denoted by $sigma_1(\Omega)$. Note that, the norm of the observation matrix is strongly correlated with $NMAE$; this fact has been justified in Figure 8 and Table 5. Also, Tables 3 and 4 shows the NMAE and the processing gains (indicated by the Time Gain) achieved by minimizing the norm of the observation matrix using both GA and PSO algorithms.

Due to some inconsistency in Table 3 this part is ignored, representing the fact that to achieve a better perfromance the ultimate performance metric must be optimized.

As we have seen in the previous results, the SNIPER can improve the estimation accuracy under hard constraints of measurement resources. However, since the genetic algorithm targets the ultimate matrix completion inference performance, the processing power is problematic in large-scale networks and this limits the scala-

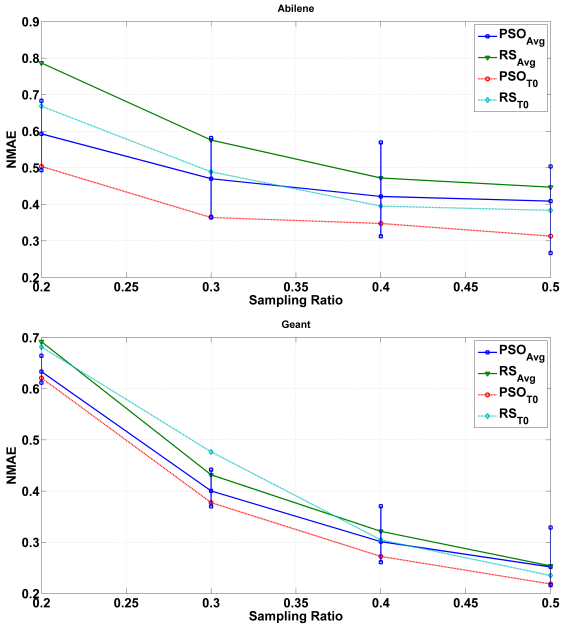

Figure 7: $NMAE$ v.s. sampling ratios.

Net./Prm.	SR	RS_{T_0}	GA_{T_0}	PSO_{T_0}	$RS^{\sigma}_{T_0}$	$GA^{\sigma}_{T_0}$	$PSO^{\sigma}_{T_0}$
Abilene	0.3	0.499312	0.364073	0.387655	0.498732	0.503375	0.480801
GEANT	0.2	0.728394	0.55408	0.60432	0.722275	0.679437	0.684696

Net./Prm.	SR	RS_{Avg}	GA_{Avg}	PSO_{Avg}	RS^{σ}_{Avg}	GA^{σ}_{Avg}	PSO^{σ}_{Avg}
Abilene	0.3	0.569638	0.556108	0.494916	0.582961	0.546438	0.586098
GEANT	0.2	0.686854	0.601478	0.644993	0.69442	0.664646	0.681144

Table 3: NMAE comparison between different evolutionary fitness functions (σ denotes norm function).

Net./TimeGain	SR	$TimeGain^{\sigma}_{GA}$	$TimeGain^{sigma}_{PSO}$
Abilene	0.3	0.937689	0.818912
GEANT	0.2	0.884593	0.773611

Table 4: Time Gain comparison between different evolutionary fitness functions (σ denotes norm function).

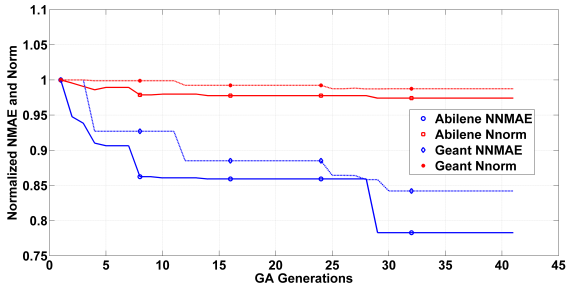

Figure 8: The evolution of normalized $NMAE$ and norm in the GA.

	$SR = 0.2$	$SR = 0.3$	$SR = 0.4$	$SR = 0.5$
ρ Abilene	0.9089	0.9640	0.7546	0.9698
ρ Geant	0.9337	0.8805	0.9617	0.8791

Table 5: Correlation coefficient $\rho := \frac{Cov(NMAE,\sigma_1(\Omega))}{\sqrt{Var(NMAE)Var(\sigma_1(\Omega))}}$ between NMAE and the norm of the observation matrix

bility of the SNIPER framework. To reduce the computational complexity and processing power of the SNIPER two approaches are considered here. First, instead of minimizing the ultimate estimation accuracy (e.g. represented by $NMAE$), the norm of the observation matrix Ω is minimized where the norm of a matrix is defined as the maximum singular value of the matrix. Note that, the norm of the observation matrix is strongly correlated with $NMAE$; this fact has been justified in Figure 8 and Table 5. Second, instead of using the GA we use the PSO evolutionary optimization algorithm which is much faster than the GA and can reduce the computational complexity of the SNIPER and accordingly reduce the processing power.

5.4 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 [28].

6. CONCLUSION

In this paper

7. REFERENCES

[1] A. Clemmi, "Network management fundamentals," *Cisco Press*, 2006.

[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] 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.

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

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

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

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

[10] 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.

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

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

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

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

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

[16] L. Ma, T.He, K. Leung, A. Swami, and D. Towsley, "Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement," *IEEE/ACM Transactions on Networking*, 2014.

[17] L. Ma, T.He, K. Leung, A. Swami, and D. Towsley, "Monitor placement for maximal identifiability in network tomography," *Proc. of IEEE INFOCOM*, 2014.

[18] X. Wang, M. Malboubi, C. Chauh, and S. Wang, "Practical approach to identify additive link metric with shortest path routing," *To be submitted to IEEE, Communication Letters*, 2014.

[19] M. Malboubi, Y. Gong, W. Xiong, C. Chuah, and P. Sharma, "Software defined network inference with passive/active evolutionary-optimal probing (sniper)," tech. rep., 2014. At: `http://www.ece.ucdavis.edu/cerl/techreports/2014-2/`.

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

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

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

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

[24] 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.

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

[26] "Abilene traffic." www.cs.utexas.edu/ yzhang/research/AbileneTM.

[27] "Geant network:." http://totem.info.ucl.ac.be/dataset.html.

[28] "Miniet," At: `http://mininet.org/`.