# 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.

## 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

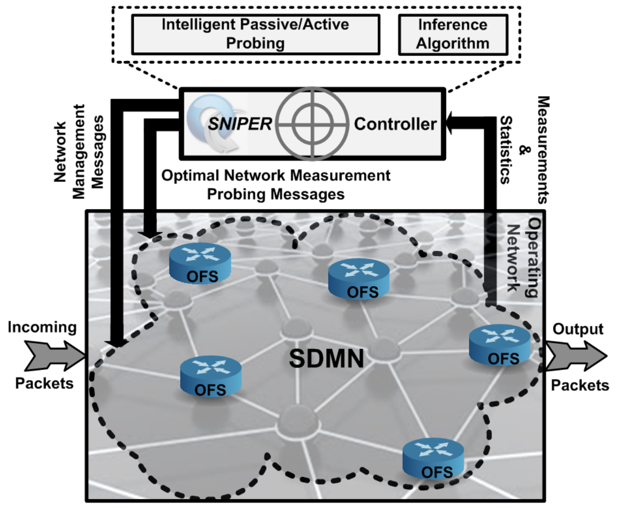
Intelligent Passive/Active Probing
Inference Algorithm
SNIPER
Controller
Network Management Messages
Measurements & Statistics
Optimal Network Measurement Probing Messages
Operating Network
OFS
SDMN
Incoming Packets
Output Packets

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

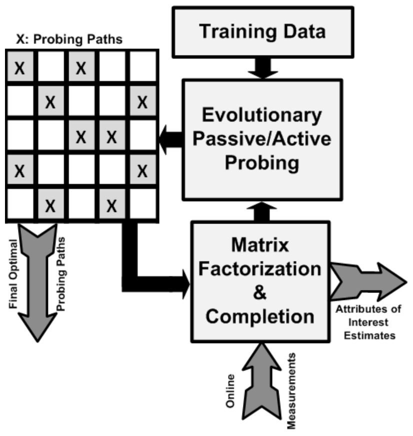
X: Probing Paths
Training Data
Evolutionary Passive/Active Probing
Matrix Factorization & Completion
Final Optimal Probing Paths
Attributes of Interest Estimates
Online Measurements

*Figure 2:* Evolutionary optimal network probing process.

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 OMP messages include passive/active probing signals which exactly indicate which IAI must be accurately measured at different times or spaces.

As it is shown in Figure 2, the optimal network probes are computed in the learning stage and 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-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

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

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 $\Omega$ is the set of observed (i.e. directly measured) entries, $\bar{\Omega}$ is the complement of $\Omega$ (which identifies missed or unobserved entries), $P_\Omega$ is a sampling function that preserves entries of $X$ in $\Omega$ (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 $\Omega$. 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 $\hat{X} = U_{n\times r}V_{r\times n}^T$ and $\lambda$ 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 spatio-temporal redundancies that can be used to estimate non-observed or missed entries.

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 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.

## 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 opti-

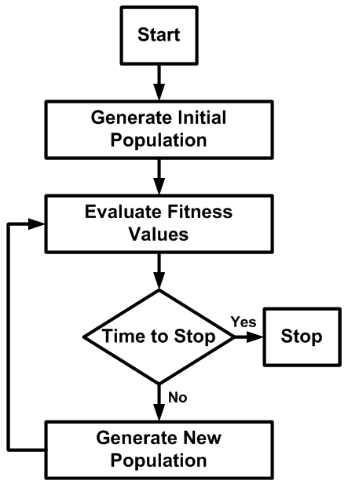

*Figure 3:* Evolutionary optimization algorithm's flowchart

mization 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, 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) and $\hat{x}_{ij}$ denotes the $ij^{th}$ entry of the matrix $\hat{X}$ which is output of the MC process.

$$\begin{aligned} NMAE &= \frac{\sum_{ij\in\bar{\Omega}} |x_{ij} - \hat{x}_{ij}|}{\sum_{ij\in\bar{\Omega}} |x_{ij}|} \\ NMSE &= \frac{\sqrt{\sum_{ij\in\bar{\Omega}} (x_{ij} - \hat{x}_{ij})^2}}{\sqrt{\sum_{ij\in\bar{\Omega}} (x_{ij})^2}} \end{aligned} \tag{3}$$

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.(3). Accordingly, the best chromosomes, with lowest fitness values, are selected and the crossover operation (as defined in Eq.(4)), 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.

$$\begin{aligned} \text{OffSpring}_1 &= C_1(1:r,:) + C_2(r+1:n,:) \\ \text{OffSpring}_2 &= C_2(1:r,:) + C_1(r+1:n,:) \end{aligned} \tag{4}$$

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.(3)) among all iterations (for one particle) or among all particles.

The velocity $V_i^k$ is updated according to Eq.(5) where $c_1$ and $c_2$ are acceleration constants, which here they setup to $c_1 = c_2 = 2$, and $\alpha_1$ and $\alpha_2$ are standard uniform random variables in interval $[0, 1]$. The positive inertia weight $\omega$ is computed as $\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{k}{N_i}$ where $\omega_{min}$ and $\omega_{max}$ 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 over $N_i$ 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).

$$V_i^k = \omega V_i^{k-1} + c_1\alpha_1(BP_i^k - P_i^{k-1}) + c_2\alpha_2(GP^k - P_i^{k-1}) \tag{5}$$

## 4. 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.