Non-negative Matrix Factorization For Network Delay Matrix Completion

Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton, Thierry Chonavel

Non-negative Matrix Factorization For Network Delay Matrix Completion

Sanaa Ghandi, Alexandre Reiffers-Masson, Sandrine Vaton, Thierry Chonavel
IMT Atlantique, LAB-STICC laboratory
Brest, France
{sanaa.ghandi, alexandre.reiffers-masson, sandrine.vaton, thierry.chonavel}@imt-atlantique.fr

***Abstract*—Accurate estimation of delays in a network is crucial for its management. In real-world applications, it is not always possible to conduct on-demand measurements regularly on the overall network. Doing so is costly and time-consuming, and it is also possible that not all the equipments respond to the probes sent in the network. In this paper, we formulate the network delay prediction problem as a non-negative matrix factorization problem with piecewise constant coefficients of the approximate instantaneous representation of data. We choose this approach to utilize the strong spatial and temporal correlation that appear in network delay data. To solve this factorization problem, we consider two different algorithms: an alternating projected gradient algorithm and the NeNMF algorithm. We finally study the efficiency of our approach on two datasets. The first dataset is a synthetic dataset produced by a simulator that we have designed, and the second one is composed of RTT measurements from RIPE Atlas.**

***Index Terms*—Non-negative Matrix Factorization, Matrix Completion, Network Measurement, Network Delay**

I. Introduction

Internet delay measurements have wide applications and are crucial for proper network monitoring. They are used to measure network QoS. Moreover, in distributed services, such as content distribution network and overlay routing, knowledge of delays helps to improve responsiveness by choosing the most suitable communication peer. A full knowledge of network delays might sound attractive but conducting on-demand measurements regularly on the overall network is costly and time-consuming. It is therefore necessary to be able to predict missing delays from few measurements.

This prediction problem can be solved because delays are highly correlated. Indeed, delay measurements between different origins and destinations are spatially correlated with each other. In addition, if one considers a particular pair of nodes, the delay is generally stable over long periods of time, resulting in temporal correlation. These spatial and temporal correlations can be exploited to reduce the number of measurements. One natural way to use these correlations is to formulate our delay inference problem as a matrix completion problem [1].

We define the network delay prediction problem as a piecewise constant non-negative matrix factorization to incorporate expert knowledge into the completion problem [2]. Firstly, the non-negative constraint on the coefficients of the factorized matrices is consistent with the fact that network delays cannot be negative. Secondly, we are looking for a piecewise constant factorization to encode the temporal correlation.

978-1-6654-0601-7/22/$31.00

A. Related works

Network delays estimation has first been addressed in the context of network coordinate systems using either Euclidean embedded models [3], [4] or matrix factorization [5]–[7]. Euclidean embedded approaches are embedding network's nodes in a low-dimensional space. In this space, a good approximation for the delay between nodes is assumed to be equal to the Euclidean distance. Such approaches decrease the amount of end-to-end delays needed for the estimation of distances. However, their major drawbacks are that they are limited by geometrical constraints (triangle inequality, symmetry), and they are not efficient in the presence of complex routing policies [8].

To overcome this problem, matrix factorization approaches gained a lot of interest. These methods are often based on the low rank approximation of delay matrix. Many models use nuclear norm for rank minimization and others use non-negative matrix factorization (NMF) exploiting the positivity of delays [9], [10]. Matrix and tensor completions have also been recently used to estimate the traffic matrix [11]–[13] and to infer top-k Elephant flows [14].

The estimation of network delays with completion approaches has been studied in multiple recent works [15]–[17]. An adaptive completion algorithm has been proposed in [15] to estimate network delays. Moreover, in [16], authors have studied an efficient probing strategy of (Origin, Destination) pairs to improve the performance of completion. Finally, the effect of graph-Laplacian regularization on the performance has been studied in [17]. Such regularization is used to add more non-linear interactions between network delays.

B. Organization of the paper

To the best of our knowledge, our paper is the first to model network delay prediction problem as a piecewise constant non-negative matrix factorization problem. The code and the data used in this paper are available at [18]. Our paper is divided into 4 sections. In section II, we introduce the matrix completion approach proposed to predict network delays. We formulate the problem as the optimization of a quadratic function with a regularization term. The goal of the

regularization term is to capture the fact that delay is stable over long periods of time. Then, in Section III we introduce two algorithms used to solve the optimization problem: an alternating projected gradient algorithm [2], and the NeNMF algorithm [19] which is based on the Nesterov method. In Section IV, we first study the performance of our approach on a synthetic dataset. The simulator designed to generate the synthetic dataset is described into details. We finally assess the performance of delay completion on RIPE Atlas [20] measurements. In Section V, we conclude our paper and discuss possible extensions.

II. Network delay prediction by Non-negative Matrix Factorization

A. Non-negative matrix factorization of delays

We consider that time is slotted with time slots of ~ 5 minutes. A delay matrix over a network of N nodes at a given instant t is a $N \times N$ matrix $X(t)$ where $X_{ij}(t)$ is the delay between the origin node i and the destination node j at instant t. The delay matrix has many characteristics, including positive entries and low effective rank. The low effective rank property in this case results from spatial correlation. Indeed, some nodes, such as nodes in the same Autonomous System, often share route segments and therefore exhibit similar performance. Moreover, successive delays between a same pair of origin and destination are correlated through time since delay is stable over long periods of time on a particular path. For instance, Figure. 1 displays the time series of Round Trip Times (RTT) between two anchors on RIPE Atlas [20] during one week. We can observe that the delay is stable for hours and abrupt changes in the statistical distribution of the delay occur due to routing changes (see for example [21]). In order to take advantage of this property, we will construct a matrix D where the t-th column is given by $D_t = vec(X(t))$. Therefore, a row of D represents a (Origin, Destination) pair, and a given column represents an instant. The matrix D is of size $n \times m$ where n is the number of node pairs, and m is the number of time slots. The temporal stability of the rows of this matrix represents a low rank property. Which is indeed sufficient for a proper matrix factorization [22].

B. Optimization problem

Non-negative matrix factorization (NMF) consists in approximating the matrix D as the product of two low rank matrices $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$ with $r \leq min(n, m)$. To derive this approximation, we solve the following constrained optimization problem: $\min_{W,H} \|D - WH\|_F$ subject to $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$ where $\|.\|_F$ is the matrix Frobenius norm.

If $D_{\cdot j} = (D_{ij})_{1\leq i\leq n}$ denotes the column vector of delays at time j then the basic idea behind non-negative matrix factorization is that $D_{\cdot j} \simeq \sum_{k=1}^{r} W_{\cdot k} H_{kj}$ with $W_{\cdot k}$ the k^{th} column of matrix W. The columns of W are the basis vectors of the decomposition of the delays, and H_{kj} are the weights of this decomposition at time j.

In real-world applications, network performance is often only partially monitored (in time or space). This fact implies that in practice, only part of the elements of D are observed. Moreover, it is possible that some equipments are not replying to the probes, and therefore, are producing extra missing values in D.

The idea behind matrix completion is to approach D solely on the basis of available measurements. We introduce a $n \times m$ binary matrix S, named the sampling matrix, where $S_{ij} = 1$ if D_{ij} is known and $S_{ij} = 0$ otherwise. The optimization criterion then becomes: $\min_{W,H} \|S \circ (D - WH)\|_F$ subject to $W \in \mathbb{R}_+^{n\times r}$ and $H \in \mathbb{R}_+^{r\times m}$, where $\circ$ is the Hadamard product (term by term product of matrices).

In order take advantage of the time stability observed among the column of D, we add an additive regularization term $\beta \sum_{i=1}^{r} \sum_{j=2}^{n} |H_{ij} - H_{i(j-1)}|$ to $\|S \circ (D - WH)\|_F{}^2$. This permits to favor solutions such that the delays are more stable over time. β is a weighting hyperparameter that controls temporal smoothness.

We then arrive at the optimization problem:

$$\min_{W,H} C(W, H) \text{ s.t. } W \geq 0, H \geq 0, \tag{1}$$

where the optimization criterion can be splitted into two terms:

$$\begin{array}{lll} & C(W,H) & = F(W,H) + L(H), \\ \text{with} & F(W,H) & = \|S \circ (D - WH)\|_F{}^2, \\ \text{and} & L(H) & = \beta \sum_{i=1}^{r} \sum_{j=2}^{n} |H_{ij} - H_{i(j-1)}|. \end{array} \tag{2}$$

III. Solution of NMF factorization

In this section, we are going to introduce two algorithms that can be used in order to solve the optimization problem (1). The first one is an alternating projected gradient algorithm [2]. And the second one involves Nesterov's optimal gradient method [19].

A. Alternating projected gradient

The alternating projected gradient algorithm is an iterative algorithm where estimates of W and H are updated sequentially. Each iteration k can be decomposed into two steps: i) a steepest descent method is used to update W with $H = H^k$ fixed and with a projection over the set $W \geq 0$, and ii) a steepest descent method is used to update H with $W = W^{k+1}$ fixed and with a projection over $H \geq 0$:

$$\begin{array}{l} W^{k+1} = [W^k - \alpha \nabla_W F(W^k, H^k)]_+ \\ H_{ij}^{k+1} = [H_{ij}^k - \alpha [\nabla_H F(W^{k+1}, H^k)]_{ij} + \frac{\partial L(H)}{\partial H_{ij}}]_+, \forall (i,j) \end{array} \tag{3}$$

With $[x]_+ = max\{0, x\}$ and $\alpha \geq 0$ is a step-size. The mathematical closed expression of the first order derivatives $\nabla_W F(W, H)$, $\nabla_H F(W, H)$ and $\frac{\partial L(H)}{\partial H_{ij}}$ are:

$$\begin{array}{l} \nabla_W F(W,H) = S \circ ((WH - D)H^\top) \\ \nabla_H F(W,H) = W^\top (S \circ (WH - D)) \\ \frac{\partial L(H)}{\partial H_{ij}} = \beta(\text{sign}(H_{ij} - H_{i(j+1)}) - \text{sign}(H_{i(j-1)} - H_{ij})) \end{array} \tag{4}$$

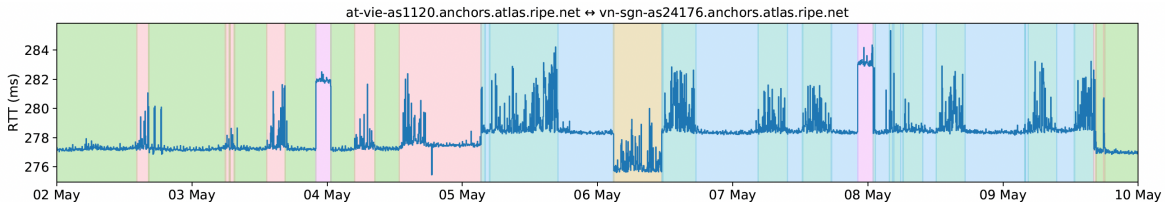

Fig. 1. RTT between two anchors on RIPE Atlas over 1 week

B. NeNMF: Nesterov gradient

The NeNMF overcomes NMF solvers limitations such as numerical instability, slow convergence and theoretical convergence problems. It achieves the optimal convergence rate $\mathcal{O}(\frac{1}{k^2})$ by using the Nesterov accelerated gradient [23]. The NeNMF consists of updating at each outer iteration k sequentially W^k and H^k by running an inner loop of Nesterov accelerated gradient method to minimize the objective function with respect to W with H^k fixed (and vice versa). In fact, in order to estimate H^k, Nesterov gradient method will construct two sequences $\{H_t\}$ and $\{Y_t\}$ and update them respectively at each iteration round t :

$$\begin{array}{l} H_t = \min_{H\geq 0}\{\phi(Y_t, H) = F(W^k, Y_t) \\ +\langle \nabla_H F(W^k, Y_t), H - Y_t\rangle + \frac{L_c}{2}\|H - Y_t\|_F{}^2\} \\ Y_{t+1} = H_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(H_t - H_{t+1}), \end{array} \quad (5)$$

where $H \mapsto \phi(Y_t, H)$ is a quadratic majorant function of $H \mapsto F(W^k, H)$ at Y_t [24]. $\langle\cdot,\cdot\rangle$ is the matrix inner product, $L_c = \|W^\top W\|_2$ is a Lipschitz constant of the gradient of the objective function $H \mapsto F(W^k, H)$ and Y_t is a linear combination of the two last approximate solutions, i.e. H_{t-1} and H_t. The coefficient α_t is updated at each iteration according to the formula $\alpha_{t+1} = (1 + \sqrt{4(\alpha_t)^2 + 1})/2$.

When solving the first-order optimality conditions (KKT conditions) for the convex optimization problem (5), the previous equations can be rewriten as:

$$\begin{array}{ll} \text{Step 1:} & H_t = [Y_t - \frac{1}{L_c}\nabla_H F(W^\top, T_k)]_+, \\ \text{Step 2:} & Y_{t+1} = H_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(H_t - H_{t+1}). \end{array} \quad (6)$$

This algorithm doesn't use the penalization $L(H)$, and therefore, in the rest of the paper, when we refer to the NeNMF algorithm, we implicitly assume that $\beta = 0$ (i.e. $C(W, H) = F(W, H)$).

IV. Performance evaluation

A. Synthetic dataset generation

To start our study, we work on synthetic data. It will provide a ground truth that will be used to compare the results of the completion algorithms and permits to rely on a fully controlled data generation environment.

There are some characteristics in actual network delay measurements that we want to reproduce. As it can be noticed in Figure 1, delay is stable over several hours and abrupt changes occur. The statistical distribution of the delay during the stable periods displays a baseline and some random variations above this baseline. Indeed, the delay in a network has several components. A term corresponds to signal propagation and packet processing at the routers' interfaces. Another term corresponds to the waiting phenomenon. The baseline can be explained by the first term, and the random component of the delay by queuing (which depends on the traffic load in the network).

Moreover, the delays between different (Origin, Destination) pairs of nodes are sometimes correlated with each other. Indeed, part of the paths between origins and destinations are shared which creates spatial correlation.

Our delay simulator is based on an explanatory model. Let us consider the case of a single Autonomous System (AS) which is modeled as a graph where the nodes of the graph are the routers. Some nodes can be the origin or destination of traffic. The Origin Destination (OD) traffic matrix represents the traffic demand (in bytes) between each origin node and each destination node. To generate this traffic matrix, we use a simple gravity model [25].

The gravity model assumes that the traffic between node i and node j is proportional to the product of two terms, one of them representing the proportion of traffic entering through origin node i, and the other representing the proportion of traffic which exits through the destination node j. So, $T_{(i,j)} = T \times y_i^{(o)} \times y_j^{(d)}$ where $T_{(i,j)}$ is the traffic between nodes i and j and T is the total traffic demand. $y_i^{(o)}$ and $y_j^{(d)}$ are proportions so that $\sum_i y_i^{(o)} = \sum_j y_j^{(d)} = 1$ and $T = \sum_{i,j} T_{(i,j)}$.

To be more precise, time is slotted so that $T_{(i,j)}(k)$ (respectively $T(k)$) represent the traffic between nodes i and j (respectively, the total traffic demand) in a time window k (lasting a few minutes). Similarly, the proportions $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$ depend on k. The total traffic demand $T(k)$ may not be constant but may be a cyclical term to take into account daily variability.

Once the OD traffic matrix has been generated, we can deduce the traffic volumes on the links of the network. If one considers a particular link l then the traffic $T^l(k)$ through link l is the aggregation of all the origin destination demands which routes go through link l. So, $T^l(k) = \sum_{\{(i,j), l\subset P(i,j)\}} T_{(i,j)}(k)$ where $P(i,j)$ is the path from i to j. In our simulator we have assumed that routes follow the shortest paths.

The delay from node i to node j is then obtained as a sum of delays along the path $P(i,j)$. A delay is associated

with each of the links using an $M/M/1$ queue model. In an $M/M/1$ model the average delay of a link l with capacity C_l and offered traffic $T^l(k)$ is equal to $\frac{1}{\mu}\frac{1}{1-\rho}$. Where $\frac{1}{\mu}$ is the average service time, that is to say the packet size divided by the bandwith of the router interface, and does not depend on the offered traffic. $\rho = \frac{T^l(k)}{C_l}$ is the load (and $\rho < 1$ since we assume that the queue is stable). Then, in the simulator the delay $D_{ij}(k)$ from node i to node j is modeled by an M/M/1 and given by $D_{ij}(k) = \sum_{l, l \subset P(i,j)} \frac{\text{Packet Size}}{C_l} \frac{1}{1-\frac{T^l(k)}{C_l}}$

Finally, our goal is to simulate realistic time series, taking into account in particular the spatial correlation but also the temporal correlation of the delays. In particular, we want to simulate abrupt changes in the statistical distribution of delays as shown in Figure 1. In the case of an AS such abrupt changes can be due to modifications of the external routing, which leads to changing the entry or exit points of part of the traffic.

So, in our simulator the terms $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$ of the gravity model are not constant. They remain constant for a while, and then, according to Markovian dynamics, a change of state occurs. As the state of the Markov chain changes, the values of the proportions $y_i^{(o)}(k)$ and $y_j^{(d)}(k)$ change, while respecting the normalization constraints.

Figure. 2 displays some delay timeseries obtained from the simulator. It can be observed that, the signals present abrupt changes, some of which are synchronized between several (Origin, Destination) pairs, with a baseline delay value over stable periods. For the simulations we considered a network of 150 nodes with 8 origins and 8 destinations of traffic ($n = 64$ (Origin, Destination) pairs).

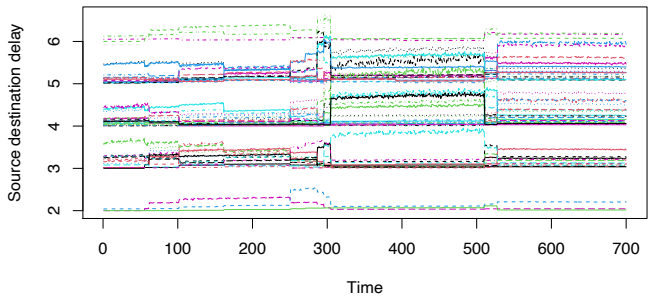

Fig. 2. Simulated delay timeseries

B. Matrix completion results

The matrix D, is obtained by vectorizing the (Origin, Destination) delays and by concatenating the successive values of these vectors of delays. The low-rank property can be justified by the strong decreasing attitude of its singular values. In fact, the 5 greatest of the 64 singular values are 409.13, 10.12, 5.45, 3.65 and 2.45. So a small value of r is sufficient to capture most of the energy of the matrix D.

To evaluate the performance of matrix factorization algorithms we consider the convergence stress [26]. It measures the quality of reconstruction of the missing values D_{ij} (i.e. such that $S_{ij} = 0$). It is defined as: $P^k_{\text{stress}} = \sqrt{\frac{\sum_{i,j}(1-S_{ij})(D_{ij} - [W^k H^k]_{ij})^2}{\sum_{i,j}(1-S_{ij})(D_{ij})^2}}$, where k is the iteration number in the matrix factorization algorithm.

Figure 3 represents the evolution of the convergence stress of the alternating projected gradient algorithm as a function of the number of iterations. We consider the first 100 and 10000 iterations and two different values $\beta = 0$ and $\beta = 0.4$. The sampling rate is 0.7 that is to say that 70% of the delay values D_{ij} are known (i.e. $S_{ij} = 1$) and 30% of the delays are supposed to be unknown.

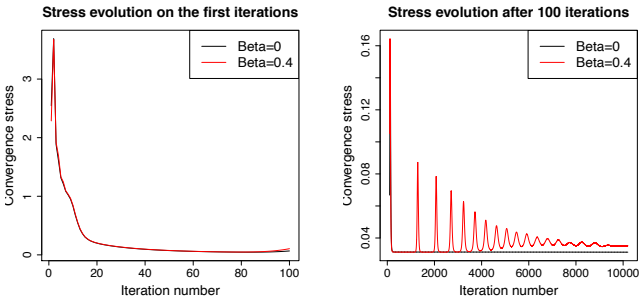

Fig. 3. Evolution of the convergence stress over the first 100 and 10000 iterations of the alternating projected gradient algorithm.

We can notice that the stress decreases rapidly in the first iterations. But one can also notice oscillations of the stress value over the long term when $\beta \neq 0$. These oscillations are due to the choice of an $L1$ norm in the penalty term $L(H)$. Indeed the derivative $\frac{\partial L(H)}{\partial H_{ij}}$ takes only three values $(2\beta, 0, -2\beta)$ and there are probably some abrupt changes in the value during iterations. A $L2$ norm would probably lead to a smoother evolution of the stress.

We are also interested in the impact of rank r on the quality of the reconstruction. Table I gives the stress values of the alternating projected gradient (APG) once the algorithm has converged (after 150.000 iterations) and of the NeNMF (after 10.000 iterations). We considered $\beta = 0$ and a percentage of missing data of 30%.

Rank r	1	2	3	4	6	8	10
Stress APG	0.031	0.031	0.014	0.011	0.007	0.005	0.003
Stress NeNMF	0.031	0.021	0.016	0.013	0.009	0.007	0.007

TABLE I
IMPACT OF RANK r ON THE STRESS.

We have also assessed the influence of the sampling rate. For values ranging from 50% to 95% of delays known (i.e. $S_{ij} = 1$), Table II gives the value of the stress after 150000 iterations of alternating projected gradient (APG) , 10000 iterations of the NeNMF method and for $\beta = 0$.

Table III gives the influence of β on the stress value for the APG with a 30% missing data and after 150.000 iterations. As

Sampling rate	50%	60%	70%	80%	90%	95%
Stress APG	0.020	0.015	0.015	0.014	0.014	0.014
Stress: NeNMF	0.014	0.013	0.013	0.013	0.012	0.012

TABLE II
IMPACT OF SAMPLING RATE ON THE STRESS.

we can see, the effect of a regularization term is robust with respect to the choice of β. Choosing $\beta \in \left[10^{-2}, 5.10^{-2}\right]$ yields a stress smaller than 2%. Finally, it is interesting to

β	0.001	0.01	0.05	0.1	0.3	0.5	0.7
Stress	0.017	0.015	0.015	0.019	0.018	0.017	0.019

TABLE III
IMPACT OF β ON THE STRESS.

observe the acceleration provided by Nesterov's method in the optimization of the criterion $F(W, H)$. In Figure 4, the value of the stress over the first 100 iterations of projected alternating gradient and of the NeNMF algorithm is represented. While the projected alternating gradient exhibits slow convergence and numerical instabilities, the NeNMF algorithm converges within fewer iterations. We also evaluated the execution time of the alternating gradient method over 100.000 iterations and the NeNMF method over 1000 iterations. We used a $2,6$ GHz Intel Core i7 processor with a 32Go 2667 MHz DDR4 memory. For the alternating gradient, the execution time was 16 minutes and for the NeNMF it was 472s. When considering real-world data in the following section, this execution time difference leads us to opt for the NeNMF algorithm.

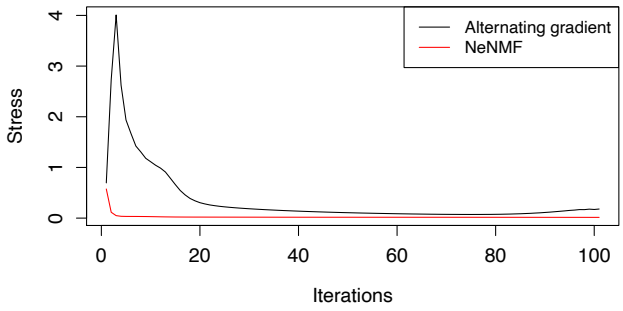

Fig. 4. Stress evolution over the first 100 iterations

C. RIPE Atlas RTT dataset

In this section, we analyze delay measurements from the RIPE Atlas platform. RIPE Atlas [20] is a global and public measurement platform held by RIPE NCC, that provides data on Internet network availability. Our dataset corresponds to delay measurements conducted each four minutes between a set of anchors chosen randomly around the world. The measurement campaign covers a period of one week. These delays were collected using three ICMP pings, and the minimum value is saved for each timestamp. This provides a dataset of 720 RTT series of 2520 time slots each.

In this dataset, the overall missing delays represent 25% of the measurements. We can observe on the heatmap Figure 5 that missing values are not distributed uniformly. Patterns vary from a (Origin, Destination) pair to another. Missing measurements may be due to a dysfunction of the origin or destination anchor, and such problems can be temporary or permanent. Only (Origin, Destination) pairs with a rate of missing data lower than 80% were kept for the experiments. Moreover, for each (Origin, Destination) time serie, we have considered as outliers the points that are above (resp.below) $\mu + 2\sigma^2$ (resp. $\mu - 2\sigma^2$) where μ is the mean of the known delay values and σ^2 is their variance. These values were removed and considered as missing delays. So, the size of matrix D is 572×2520 and the proportion of missing values is 18% (i.e. $S_{ij} = 0$). According to the previous results on the synthetic data, we decided to apply only the NeNMF on the Ripe dataset due to its speed. Since we lack ground truth for the non-measured values for this dataset, we use the following criterion to evaluate the factorization quality on observed values D_{ij}: $P^k_{\text{error}} = \sqrt{\frac{\sum_{i,j} S_{ij}(D_{ij} - [W^k H^k]_{ij})^2}{\sum_{i,j} S_{ij}(D_{ij})^2}}$, where k is the iteration number. The algorithm converges within hundreds of iterations, and we observe that the error decreases with the rank. We fix the number of iterations to 1000 and the rank to 100. In this experiment, the reconstruction error on the observed values is 2%. This low error rate is highlighted by Figure 6 which displays 5 different RTT series completed using the NeNMF. We can notice that the completed segments capture the overall baseline of the original RTT series.

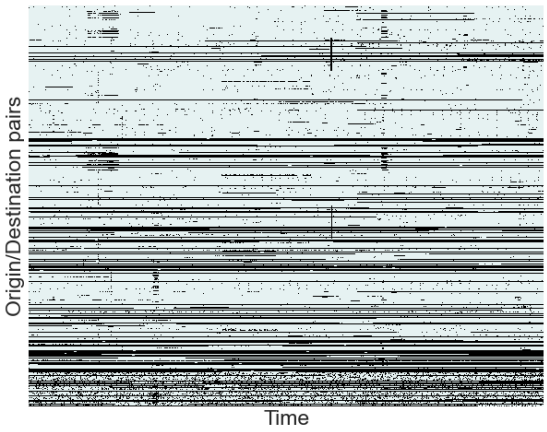

Fig. 5. Heatmap showing the missing measurements (in black) in Ripe data

V. CONCLUSION AND FUTURE WORK

Network delays are of great importance in network monitoring. Some of these measurements can be missing due to infrastructure problems, packet loss, or simply to the measurement strategy. In this paper, we addressed the problem of inferring these missing delays within a matrix completion approach. This was conceivable thanks to the stability of delays throughout time that is a contributing factor to the

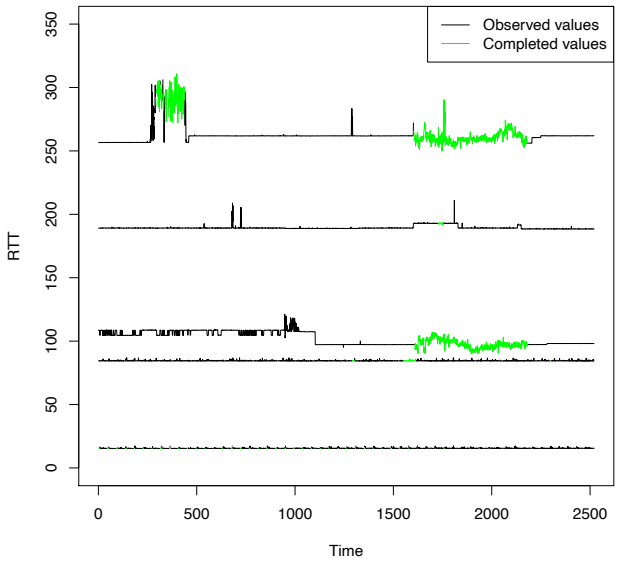

Fig. 6. RTT completion using NeNMF

low rank property of the delay matrix. We used two Non-negative matrix factorization algorithms: the alternating projected gradient and the NeNMF. We were able to test these methods in a controlled environment by using a synthetic delay generator and on real-world data with delays from Ripe Atlas. The two approaches are simple, easy to implement and show great accuracy on the completion task when applied to synthetic data. The experiments however, pointed out the speed difference between the two algorithms. The alternating projected gradient converges slower than the NeNMF. Hence, the scalability of the NeNMF was exploited by applying it to real-world dataset. The completion given by this algorithm has shown great accuracy within small number of iterations. Despite good performance demonstrated by the two approaches, the regularization term using β wasn't of interest in this context, and we think that it can be explored in future work by testing an L2-norm. This will provide more stability and can be useful for change detection in network delays. We also plan to validate NMF approaches on other real-world datasets, such as the Abilene topology, the Harvard-226 dataset, and the full anchoring mesh of Ripe Atlas. Finally, we intend to improve this work by choosing strategies more sophisticated than the uniform sampling.

REFERENCES

[1] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[2] N. Seichepine, S. Essid, C. Févotte, and O. Cappé, "Piecewise constant nonnegative matrix factorization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6721–6725.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 15–26, 2004.

[4] T. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1. IEEE, 2002, pp. 170–179.

[5] Y. Mao, L. K. Saul, and J. M. Smith, "Ides: An internet distance estimation service for large networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2273–2284, 2006.

[6] Y. Liao, W. Du, P. Geurts, and G. Leduc, "Dmfsgd: A decentralized matrix factorization algorithm for network distance prediction," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1511–1524, 2012.

[7] R. Zhu, B. Liu, D. Niu, Z. Li, and H. V. Zhao, "Network latency estimation for personal devices: A matrix completion approach," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 724–737, 2016.

[8] G. Wang, B. Zhang, and T. E. Ng, "Towards network triangle inequality violation aware distributed systems," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 175–188.

[9] Y. Mao and L. K. Saul, "Modeling distances in large-scale networks by matrix factorization," in *Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement*, 2004, pp. 278–287.

[10] L. Chai, X. Luo, F. Zhao, M. Li, and S. Liu, "Network coordinate system using non-negative matrix factorization based on KL divergence," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 2017, pp. 193–198.

[11] G. Gürsun and M. Crovella, "On traffic matrix completion in the internet," in *Proceedings of the 2012 Internet Measurement Conference*, 2012, pp. 399–412.

[12] K. Xie, C. Peng, X. Wang, G. Xie, and J. Wen, "Accurate recovery of Internet traffic data under dynamic measurements," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[13] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, and G. Zhang, "Accurate recovery of Internet traffic data: A tensor completion approach," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[14] K. Xie, J. Tian, X. Wang, G. Xie, J. Wen, and D. Zhang, "Efficiently inferring top-k elephant flows based on discrete tensor completion," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2170–2178.

[15] R. Tripathi and K. Rajawat, "Adaptive network latency prediction from noisy measurements," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 807–821, 2021.

[16] W. Meng and L. Li, "Matrix completion based adaptive sampling for measuring network delay with online support," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 14, no. 7, pp. 3057–3075, 2020.

[17] Y. Hu, L. Deng, H. Zheng, X. Feng, and Y. Chen, "Network latency estimation with Graph-Laplacian regularization tensor completion," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[18] Synthetic data generation. [Online]. Available: https://github.com/Ghandisanaa/NMF-for-network-delay-matrix-completion/tree/main

[19] N. Guan, D. Tao, Z. Luo, and B. Yuan, "NeNMF: An optimal gradient method for nonnegative matrix factorization," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2882–2898, 2012.

[20] Ripe atlas. [Online]. Available: https://atlas.ripe.net/m

[21] M. Mouchet, S. Vaton, T. Chonavel, E. Aben, and J. D. Hertog, "Large-scale characterization and segmentation of Internet path delays with infinite HMMs," *IEEE Access*, vol. 8, pp. 16 771–16 784, 2020.

[22] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward, "Completing any low-rank matrix, provably," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 2999–3034, 2015.

[23] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $mathcal\{O\}(1/k\{2\})$," in *Sov. Math. Dokl*, vol. 27.

[24] D. Seung and L. Lee, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, pp. 556–562, 2001.

[25] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 206–217, 2003.

[26] S. Liu and Q. Wang, "Poster: online adaptive sampling for network delay measurement via matrix completion," in *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 2019, pp. 1–2.