Network Anomaly Detection based on Tensor Decomposition

Ananda Streit, Gustavo Santos, Rosa Leão, Edmundo de Souza e Silva, Daniel Menasché, Don Towsley*
Federal University of Rio de Janeiro, Rio de Janeiro, Brazil *University of Massachusetts at Amherst, USA

***Abstract*—The problem of detecting anomalies in time series from network measurements has been widely studied and is a topic of fundamental importance. Many anomaly detection methods are based on packet inspection collected at the network core routers, with consequent disadvantages in terms of computational cost and privacy. We propose an alternative method in which packet header inspection is not needed. The method is based on the extraction of a normal subspace obtained by the tensor decomposition technique considering the correlation between different metrics. We propose a new approach for online tensor decomposition where changes in the normal subspace can be tracked efficiently. Another advantage of our proposal is the interpretability of the obtained models. The flexibility of the method is illustrated by applying it to two distinct examples, both using actual data collected on residential routers.**

***Index Terms*—network measurement and analysis, machine Learning for networks, DDoS detection, tensor decomposition**

I. Introduction

The problem of detecting anomalous events in computer networks has been widely studied due to its relevance to network operation. However, these events are in general very hard to identify [1]. The problem is challenging due to the wide variety of anomalies, low frequency of occurrences, and the definition of what is considered "expected behavior". An application example among the countless existing ones is detecting occasional changes in traffic patterns on a communication channel caused by a distributed denial of service (DDoS) attack. DDoS attacks represent a major threat to proper network operation, wasting resources and creating network outages. For instance, DDoS attacks targeted Amazon Web Services in October 2019 and were able to disrupt different services [2].

In general, anomaly detection is based on the analysis of packet headers at the core of the network, with potentially high computational cost and possible privacy issues. Our methodology differs from others in that it does not use packet headers; it is based on distributed data collection at home routers, and uses only a small amount of information.

The methodology is based on *tensor decomposition* to detect and diagnose anomalous events using multivariate time series. We evaluate the approach using time series obtained from measurements collected at home routers of a medium-sized ISP. Tensor decomposition allows the extraction of normal patterns from the metrics considered, during different time intervals, and the identification of latent relationships between them. We also devise a new online tensor decomposition method that efficiently tracks changes in the normal subspace. Our results show the effectiveness of the method to detect anomalies in two different scenarios used as examples. Nevertheless, we emphasize that the methodology is general and can be employed in other scenarios.

This work shares similarities with [3], [4], where a normal subspace is defined by applying PCA and model residuals are used to detect anomalies in a network. In this work, extraction of the normal subspace is performed using the PARAFAC model [5], which naturally allows the decomposition of multidimensional data and preserves relationships among the metrics under evaluation.

The first application of our approach is to detect DDoS attacks. Based on observations of byte and packet upload/download counters, which are non-intrusive and require no packet inspection, we show that the proposed method accurately detects attacks in both offline and online scenarios. Our second application is the identification of time intervals within which performance degradation occurs. (Performance degradation is the anomaly in this case.) The process is easily automated to identify and locate such anomalies and analyze the quality of service in different parts of an ISP's topology.

Contributions. Key contributions are summarized below:

- *Tensor decomposition to detect network anomalies.* Our framework is based on tensor decomposition. We show that the PARAFAC model provides an interpretable and efficient way to extract expected normal behavior, taking into account correlations among different metrics. We exemplify the application in two scenarios using different input metrics.
- *New online tensor decomposition method.* Our method is based on a *tensor window* [6]. The results show the good accuracy and efficiency of the approach.
- *Use of real data collected at home routers.* We use time series obtained from real network measurements collected at home routers to evaluate the framework. Our method is capable of detecting different types of anomalies based on simple metrics and without compromising users' privacy.

Related work is presented in Section II. The tensor decomposition technique is discussed in Section III. Section IV describes the proposed framework for anomaly detection and we explain how the residuals are extracted in both offline and online scenarios in Section V. The DDoS attack detection example application is presented in Section VI, and Section VII describes the second application example in which network performance degradation intervals are detected. Section VIII concludes our work.

II. Related work

Anomaly detection methods are based on models that capture the normal behaviour of the network [7]. Most prior work on network anomaly detection is based on packet inspection in the core of the network [3], [4], [8]–[10], which requires processing privacy-sensitive information from packet headers, such as traffic volume between source and destination IPs and port number. Recent work also employs packet inspection, but at home routers [11]. Our work uses measurements performed at home routers without packet inspection, providing a simple, efficient and privacy-preserving strategy.

Previous work by our group has also made use of measurements at home routers without packet inspection [12]. Mendonça et al. [12] focus on DDoS attacks that are detected with high accuracy using only simple byte and packet count statistics during a time window. The current work uses another method (tensor decomposition) and we show that it can be used to detect different types of anomalies. In addition, the results of our approach are easier to interpret, since PARAFAC produces interpretable models [5], and we are able to infer the normal daily behavior of a user, one of the main challenges in anomaly detection [1].

Other previous works in the literature use subspace extraction methods (like PARAFAC) to detect network anomalies. As an example, Maruhashi et al. [8] identify suspicious activities on the network (such as port scanning and spreading of worms) by searching for abnormal subgraphs from the discovered patterns returned by a tensor model (PARAFAC). Their method heavily depends on a manual choice of the patterns deemed interesting. In addition, [8] uses packet inspection, considering a dataset with format (source IP $\times$ destination IP $\times$ timestamp or port number). This data structure is commonly used for network analysis with tensor models based on packet inspection. In contrast, our work does not extract data from packet headers and considers tensors with the format (user $\times$ network metrics $\times$ timestamp).

In the works of Lakhina et al. [3], [4], the authors apply PCA to define the normal subspace. In [4] anomalies that span multiple traffic features (metrics) are detected, similar to our work. However, PCA is a matrix-based model and, unlike tensor-based models like PARAFAC, it requires multidimensional data to be unfolded [13] into a single, large matrix before its application. The PARAFAC model, on the other hand, reveals relationships between different metrics in multidimensional data, making it more robust to noise. It also has the property of uniqueness, in contrast to PCA, where its inherent rotational freedom [5] can lead to distinct interpretations concerning the structure of the normal subspace.

Xie et al. [9] proposes an anomaly detection method using a modified PARAFAC model that accounts for nonlinear data features. The proposed algorithm considers the similarity of tensor slices in each mode during the training process. In addition, residual anomalies are associated with a sparse tensor and are isolated during the optimization process. Kasai et al. [14] also proposes a sparse tensor to account for abnormal flows. However, both works ignore the interpretability of the model and evaluate the method using only artificially generated attack data taken from arbitrary probability distributions, while we consider: (i) attack traffic generated by real malware and (ii) real performance degradation events.

We propose an online tensor decomposition approach based on the sliding window method of Sun et al. [6]. Our solution incurs a lower computational cost while maintaining good performance. Kasai et al. [14] also considers a tensor-based online algorithm. Similar to our online algorithm, Kasai et al. [14] uses the concept of sliding window and modifies PARAFAC decomposition to deal with time and space complexities required for online approaches. Other works also modify PARAFAC decomposition for online application, but without the use of windows [15], [16]. We focus on a simpler online solution, with a slight adaptation in PARAFAC decomposition. In our approach the anomalies are detected by classifying or clustering the residuals obtained by tensor decomposition.

III. Tensor decomposition

In this section we briefly present tensor decomposition and describe our notation. For details we refer to [13]. A tensor is a multidimensional matrix denoted by $\mathcal{X}$. We usually refer to the dimensions of $\mathcal{X}$ as *modes*. A third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be represented by a sum of three-way outer products [13] as follows,

$$\mathcal{X} = \mathcal{M} + \mathcal{E}, \quad \mathbf{a}_r \in \mathbb{R}^I, \mathbf{b}_r \in \mathbb{R}^J, \mathbf{c}_r \in \mathbb{R}^K \tag{1}$$

$$\mathcal{M}_{i,j,k} = \sum_{r=1}^{R} \mathbf{a}_{r,i} \mathbf{b}_{r,j} \mathbf{c}_{r,k}, \tag{2}$$

where $\mathcal{E}$ is the residual tensor and R is the number of factors. The *factor matrices* (or loadings) define model $\mathcal{M}$: $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$, $B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, $C = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_R] \in \mathbb{R}^{K \times R}$. Following standard notation, we let $\mathbf{a}_r = A_{:,r}$, for $1 \leq r \leq R$, and $\mathbf{a}^{(i)} = A_{i,:}$, for $1 \leq i \leq I$.

The PARAFAC decomposition is obtained by minimizing the sum of squares of the residuals, i.e., the difference between $\mathcal{X}$ and $\mathcal{M}$. Such difference is a nonconvex function; however, if we fix two of the factor matrices, the problem is reduced to a linear least squares regression for the third matrix. This is the basis of the *Alternating Least Squares* (ALS) procedure [5]. ALS estimates the factor matrices one at a time, keeping the others fixed. The process iterates until a convergence criterion is satisfied or there is no change in estimates.

In this work we use the method of *Split-Half Validation* (SV) [17] in combination with *Tucker Congruence Coefficient* (TCC) [18] to estimate R and evaluate whether the solution is unique and generalizable.

IV. Framework

The proposed methodology consists of the following steps:

1) **Preprocessing:** In the first step we perform data transformations needed to apply tensor decomposition, such as data scaling and filtering.

2) **Tensor Decomposition:** In this step we apply tensor decomposition to extract the normal subspace. We use the PARAFAC method due to the uniqueness of its solution and its capacity to deal with multivariate data [5].

3) **Residual extraction:** The model obtained by tensor decomposition is used to extract the residuals and perform anomaly detection. The idea is that anomalies are not well modeled by the normal subspace, allowing separation between normal and anomalous behavior through residual analysis.

4) **Anomaly classification/clustering:** The final step varies depending on the application. When the dataset contains labeled anomalies, we perform a supervised classification. On the other hand, there are applications where the labels for anomalies are unknown or hard to obtain. For these cases, we consider an unsupervised approach based on clustering.

V. Residual extraction

Our anomaly detection technique is based on analyzing the PARAFAC residuals [5]. Normal behavior is captured (modeled) by tensor decomposition and anomalies are detected by investigating deviations from the modeled patterns.

A. Offline residual extraction

User data generally exhibits strong daily patterns over time. This leads us to split observations from users into independent daily series. We denote each of these series as a *User-Day pair*, or *UD pair*.

Let I denote the number of UD pairs in our dataset. We consider as inputs three-way tensors with modes UD pair (factor matrix A), metrics of interest (factor matrix B) and time (factor matrix C), denoted by indices i, j and k, respectively. Let $\mathcal{X}_{i,:,:}$ be the i-th horizontal slice of tensor $\mathcal{X}$, i.e., $\mathcal{X}_{i,:,:}$ is a two-dimensional matrix obtained by fixing the UD pair mode at value i [13]. Then, for each UD_i with measurements $\mathcal{X}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$, we obtain a model slice $\mathcal{M}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$ using PARAFAC ALS procedure. Residuals are measured as the difference between the model estimates and the input dataset $\mathcal{E}_{i,:,:} = \mathcal{X}_{i,:,:} - \mathcal{M}_{i,:,:}$, where $\mathcal{E}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$. As a UD refers to a day and our dataset consists of time series of one-minute bins, $K = 1440$.

Next, we determine the residuals corresponding to measurements from new UDs that were not previously used to parametrize model $\mathcal{M}$. Let $\tilde{\mathcal{X}}_{\kappa,:,:}$ denote the measurements corresponding to a new UD_κ. We use factor matrices B and C from the previously trained model $\mathcal{M}$ (eq. (2)) and the new measurements $\tilde{\mathcal{X}}_{\kappa,:,:}$ to obtain vector $\tilde{\mathbf{a}}^{(\kappa)} \in \mathbb{R}^{1\times R}$. Factor matrices $\tilde{A}$, B and C produce model $\tilde{\mathcal{M}}_{\kappa,:,:}$, with corresponding error $\tilde{\mathcal{E}}_{\kappa,:,:}$, where $\tilde{A}_{\kappa,:} = \tilde{\mathbf{a}}^{(\kappa)}$. Vector $\tilde{\mathbf{a}}^{(\kappa)}$ is chosen to minimize quadratic error between model estimates and measurements. Let $\tilde{\mathcal{X}}_{\kappa,:,:(1)}$ be the matrix unfolding of tensor $\tilde{\mathcal{X}}_{\kappa,:,:}$ in its first mode [13], where $\tilde{\mathcal{X}}_{\kappa,:,:(1)} \in \mathbb{R}^{1\times JK}$. Then,

$$\tilde{\mathcal{M}}_{\kappa,:,:} = \tilde{\mathcal{X}}_{\kappa,:,:} - \tilde{\mathcal{E}}_{\kappa,:,:} \Rightarrow \tilde{\mathbf{a}}^{(\kappa)}(C\odot B)^T = \tilde{\mathcal{X}}_{\kappa,:,:(1)} - \tilde{\mathcal{E}}_{\kappa,:,:(1)}$$
$$\Rightarrow \tilde{\mathbf{a}}^{(\kappa)} = \tilde{\mathcal{X}}_{\kappa,:,:(1)}((C\odot B)^T)^\dagger, \quad (3)$$

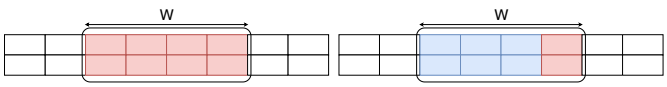
where $C \odot B$ denotes the Khatri-Rao product [13] between matrices C and B and $M^\dagger$ denotes the Moore-Penrose pseudo-inverse of matrix M [13]. Note that both $(C\odot B) \in \mathbb{R}^{JK\times R}$ and $((C\odot B)^T)^\dagger \in \mathbb{R}^{JK\times R}$. As vector $\tilde{\mathbf{a}}^{(\kappa)}$ minimizes the quadratic error, the corresponding error $\tilde{\mathcal{E}}_{\kappa,:,:(1)}$ is orthogonal to $((C\odot B)^T)^\dagger$ which implies (3). Thus, the residuals of UD_κ are obtained by $\tilde{\mathcal{E}}_{\kappa,:,:} = \tilde{\mathcal{X}}_{\kappa,:,:} - \tilde{\mathcal{M}}_{\kappa,:,:}$, where model $\tilde{\mathcal{M}}_{\kappa,:,:} \in \mathbb{R}^{1\times J\times K}$ contains the new factor vector $\tilde{\mathbf{a}}^{(\kappa)}$.

Note that the measurements corresponding to UD pairs are available by the end of a day, and residuals must be computed at that time. In addition, as network conditions may change over time, it is necessary to periodically check if $\mathcal{M}$ is still a good model (e.g., using Split-Half validation [17]). Otherwise, $\mathcal{M}$ must be retrained to compute residuals for new UD pairs.

B. Online residual extraction

The online method tracks changes in the data by continuously recomputing the model using PARAFAC. In the online scenario, time is divided into one minute slots and new data from all home routers is processed at every slot. The online decomposition considers USERS instead of UD pairs as one of the tensor modes, and obtains a three-way tensor with user (mode A), the metric of interest (mode B), and time (mode C). As soon as a new data stream arrives (every minute), the model is updated and residuals are extracted.

We consider two different online residual extraction schemes. First, we describe Full Window Optimization (FWO) [6]. Then, we propose Partial Window Optimization (PWO), a simpler and more efficient FWO variant. Figure 1 illustrates the difference between the methods, as discussed below. Note that both schemes allow expansion of modes A and B throughout online decomposition, in case new users are added or new metrics of interest are collected, respectively. The offline model $\mathcal{M}$ (resp., $\tilde{\mathcal{M}}$) denotes the model obtained before (resp., after) collecting additional measurements. In the online model, variable t already subsumes the number of collected samples, so we drop tilde from all variables.



(a) FWO (b) PWO

Fig. 1. Online tensor decomposition approaches ($W = 4$ time units). Red time slots are used to compute factor matrices A, B and C. Blue time slots are used to compute factor matrices A and B.

1) Full Window Optimization (FWO) [6]: A simple approach to online tensor decomposition is based on a *tensor window* [6] $\mathcal{X}(t, W) \in \mathbb{R}^{I\times J\times W}$ over the time mode (mode C), where W refers to the window size. At every minute t the window slides and a new tensor is formed by combining the $W-1$ previous slices $\{\mathcal{X}_{:,:,t-W+1}, ..., \mathcal{X}_{:,:,t-1}\}$ and the newly obtained data stream $\mathcal{X}_{:,:,t} \in \mathbb{R}^{I\times J\times 1}$, representing a new time slice. Since our network data presents strong daily patterns and we consider minute time slots, we define the window size $W = 1440$ mins.

To obtain models in FWO we use the same optimization method applied in the offline scenario for each sliding window.

Namely, for each window we compute a PARAFAC model $\mathcal{M}(t,W) \in \mathbb{R}^{I\times J\times W}$ using $\mathcal{X}(t,W) \in \mathbb{R}^{I\times J\times W}$ as input for the PARAFAC ALS algorithm. Residuals are measured as the difference between the model and the input dataset $\mathcal{E}(t,W) = \mathcal{X}(t,W) - \mathcal{M}(t,W)$. Usually we are interested in analyzing the behavior of the most recent sample. Therefore, we consider the residuals of the last minute t, $\mathcal{E}_{:,:,t} \in \mathbb{R}^{I\times J\times 1}$.

A good initialization for the optimization algorithm reduces the number of iterations needed to converge [6]. Therefore, to speed up convergence, after we move the window forward, we initialize the ALS algorithm with the same values used in PWO (given by lines 1-2 in Algorithm 1).

FWO requires computation of a whole new PARAFAC model at every window. As such, it may not be suitable for online applications, often requiring a large and variable number of iterations [15]. Our results indicate that this method is too computationally expensive to be used online for our application (see Figure 4). Hence, we propose a variation to decrease computational cost while providing good performance.

2) Partial Window Optimization (PWO): In order to reduce the run time we modify FWO. Consider the factor matrix related to the time mode $C(t) \in \mathbb{R}^{W\times R}$ used to model the tensor window $\mathcal{X}(t,W)$. To obtain $C(t)$ we keep the previous $W-1$ known loadings $\{\mathbf{c}(t-W+1), ..., \mathbf{c}(t-1)\}$ fixed and compute the time mode loadings related to the last sample, i.e., we compute $\mathbf{c}(t)$. The other factor matrices $A(t)$ and $B(t)$ are fully recomputed based on the tensor window $\mathcal{X}(t,W)$. Due to daily patterns in our data we define $W = 1440$, as in FWO.

Let $\mathcal{X}_{:,:,t}$ denote the measurements of a newly obtained data stream at time t. The model is estimated by updating the unknown variables ($A(t)$, $B(t)$ and $\mathbf{c}(t)$, see Figure 1(b)) alternately and iteratively, until a convergence criterion is satisfied or there is no change in estimates (Algorithm 1). As in the ALS algorithm, matrices $A(t)$ and $B(t)$ and vector $\mathbf{c}(t)$ are calculated by minimizing the quadratic error between model estimates and measurements. The sequence of updates is given by lines 4-7 in Algorithm 1.

In Algorithm 1, $\mathcal{X}_{(1)}$ and $\mathcal{X}_{(2)}$ are the tensor unfoldings of $\mathcal{X}$ in its first and second modes, respectively, $\mathcal{X}_{(1)} \in \mathbb{R}^{I\times JK}$, $\mathcal{X}_{(2)} \in \mathbb{R}^{J\times IK}$. Note that $\mathbf{c}(t) \in \mathbb{R}^{1\times R}$, $A(t) \in \mathbb{R}^{I\times R}$ and $B(t) \in \mathbb{R}^{J\times R}$. As with the FWO scheme, the factor matrices of model $\mathcal{M}(t,W)$ are initialized with the model estimates $\mathcal{M}(t-1,W)$ obtained for the previous window. The residuals related to t are obtained by

1 $A(t) \leftarrow A(t-1)$, $B(t) \leftarrow B(t-1)$
2 $C(t) \leftarrow [\mathbf{c}(t-W+1)^T, \ldots, \mathbf{c}(t-1)^T, \mathbf{c}(t-W)^T]^T$
3 **while** *not converged* **do**
4 $\quad \mathbf{c}(t) \leftarrow \mathcal{X}_{:,:,t(3)}((B(t) \odot A(t))^T)^\dagger$
5 $\quad C(t)_{W,:} \leftarrow \mathbf{c}(t)$
6 $\quad A(t) \leftarrow \mathcal{X}_{(1)}((C(t) \odot B(t))^T)^\dagger$
7 $\quad B(t) \leftarrow \mathcal{X}_{(2)}((C(t) \odot A(t))^T)^\dagger$
8 **end**
9 **return** $A(t), B(t), \mathbf{c}(t)$

Algorithm 1: Online algorithm

TABLE I
TYPES OF DDOS ATTACKS EVALUATED

Malware	Attack type (payload size)
Mirai	UDP flood (1400B)
Mirai	TCP SYN flood (0B)
Mirai	TCP ACK flood (0B)
Mirai	UDP PLAIN flood (1400B)
BASHLITE	UDP flood (1400B)
BASHLITE	TCP SYN flood (0B)
BASHLITE	TCP ACK flood (0B)

$$\mathcal{E}(t,W)_{:,:,t} = \mathcal{X}(t,W)_{:,:,t} - \mathcal{M}(t,W)_{:,:,t}.$$

VI. APPLICATION I: DDOS ATTACK DETECTION

We apply our framework to detect DDoS attacks originated from home devices. We consider a dataset with different types of attack vectors (see Table I) obtained by combining home users traffic and attack traffic measured in laboratory experiments using real malware code. Then, we apply a supervised approach to detect attacks.

A. Preprocessing

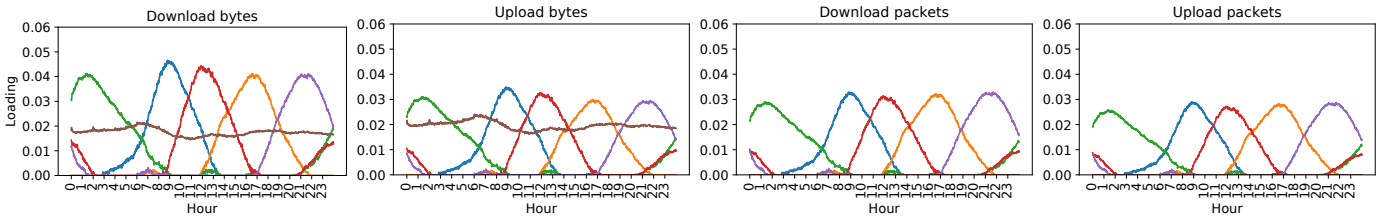
We collect per minute, per user upload and download byte and packet rates in a given day (i.e., UD pair). From this data we obtain a multivariate time series that is input to the tensor decomposition method. We consider 18722 time series from 812 users between 19-August-2019 and 22-September-2019.

We obtain our attack dataset using the approach proposed in [12]. A brief description of the methodology follows. First, we randomly choose a fraction of infected homes $q = 0.05$ that participate in synchronized DDoS attacks, where q is chosen based on the fraction of users affected by a real attack [19]. Next, we define the attack type (Table I) uniformly at random. The majority of attacks lasts for a few minutes [20]. Therefore, we consider attacks whose duration follows a Gaussian distribution with mean $\mu = 2$ minutes. We then sample time slots where the synchronized attacks start using a uniform distribution with one attack per day on average. (Similar results were obtained when different attack rates were considered.) Finally, we add the attack traffic to the measured traffic of the infected homes.

We split our dataset into three different sets. The first set (Tr_1) contains the first week of the dataset and is used to extract the normal subspace. We use a second set (Tr_2) with the following 19 days to fit an anomaly classifier using residuals extracted from the tensor model. A third set (*Te*) with the last 9 days of the dataset is used to evaluate the classifier performance.

In a real-world scenario, it is often difficult to define precisely whether a traffic dataset hides embedded network anomalies, especially in the case of malicious anomalies [4]. Therefore, we evaluate the robustness of our method by applying it to the dataset after the addition of some infected users. The idea is to consider a more realistic scenario, where some hidden anomalies may be present.

Before applying tensor decomposition, we convert our data to logarithmic scale. Then, we apply Min-Max normalization



(a) (b) (c) (d)

Fig. 2. Offline: Factors obtained by PARAFAC.

to each traffic metric, where the minimum and maximum values are taken from the training set Tr_1 and applied to the entire dataset. By keeping traffic metrics within a controlled range, we capture the correlations between them and ensure that they have the same impact on the optimization process.

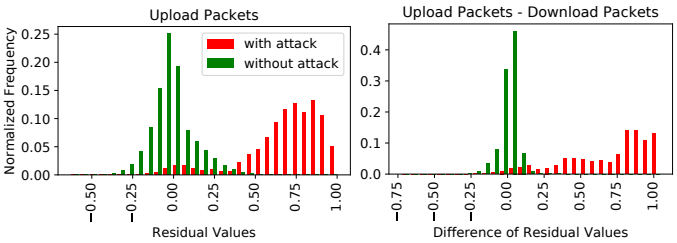
B. Tensor decomposition

We consider different tensor structures for offline and online scenarios. In the offline approach our tensor is composed of three modes: (UD $\times$ traffic metric $\times$ minute). We obtain model $\mathcal{M} \in \mathbb{R}^{3412\times4\times1440}$ from the training set Tr_1 with a total number of UDs equal to 3412. The application of *Split-Half Validation* validates up to $R = 6$ factors. (Except as otherwise noted, we use $R = 6$.) In the online approach we consider as modes (user $\times$ traffic metrics $\times$ minute), so the models $\mathcal{M}(t, W) \in \mathbb{R}^{812\times4\times1440}$ are obtained with a window size $W = 1440$ over the whole dataset, that has a total of 812 users. We consider four metrics: download and upload bytes/packets at every minute by home users.

We analyze the factors obtained in the offline scenario (model $\mathcal{M}$) to understand model behavior. Figures 2(a) and 2(b) present the time mode (mode C) factors weighted by the loadings associated with download and upload byte rate measurements (mode B), respectively, while Figures 2(c) and 2(d) show the factors weighted by download and upload packet rate loadings. One factor (represented in gray) is nearly constant throughout the day. The remaining factors identify higher network usage at different periods of the day. Moreover, the difference in scale between the number of downloaded and uploaded bytes is larger than the difference between the number of downloaded and uploaded packets. This indicates that connections exchange a similar number of download and upload packets, but upload packets usually carry less data.

C. Residual extraction

We extract residuals for sets Tr_2 and Te using the residual extraction techniques described in Section V. These residuals consist of all traffic metrics for each minute and for each UD/user (offline/online) and are used as inputs to the classifiers. The relationship between upload and download traffic can also be an important feature to detect attacks [12]. Therefore, we also consider two additional features that express the residual difference between upload and download packets and bytes, totaling six features: (i) *download bytes*, (ii) *upload bytes*, (iii) *download packets*, (iv) *upload packets*, (v) *difference between upload and download bytes* and (vi) *difference between upload and download packets*. Figures 3(a) and 3(b) present the histograms of the residuals $\tilde{\mathcal{E}}$ retrieved from the offline method for the features (iv) *upload packets* and (vi) *difference between upload and download packets*. The histograms show that the selected features satisfactorily separate residuals with and without attacks.



(a) (b)

Fig. 3. Offline: Histograms of residuals.

Figure 4 shows the run time for both PWO and FWO online methods for each minute of a day. The red dotted line indicates when a DDoS attack occurred in that period. We compute both models using a PowerEdge R230 server with a Intel Xeon E3-1220 v6 of 3Ghz with 4 cores and 64 GB of RAM. Since FWO recomputes all loadings of factor matrix $C(t)$ (time) at every slide of the window (at every minute), the time in seconds needed to recompute $\mathcal{M}(t, W)$ varies depending on data $\mathcal{X}(t, W)$, and can be large enough such as to be inadequate for an online approach. On the other hand, PWO consistenly requires smaller computational times than FWO.

D. Anomaly classification

After extracting the residuals we train a classifier to detect when an attack occurs. To estimate the method's ability to detect attacks we consider five different classifiers, leveraging features extracted using PARAFAC: *Logistic Regression*, *Decision Tree*, *Random Forest*, *Gaussian Naive Bayes* and *Multi-layer Perceptron*. We select the classifier with the best weighted F1 score in a 5-fold cross validation on the residuals obtained from the training set Tr_2. We also consider PCA

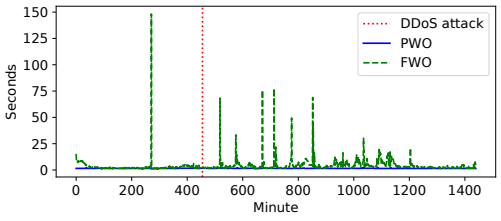


Fig. 4. Run time for FWO and PWO online methods.

TABLE II
DETECTION ACCURACY AND PRECISION WITH RANDOM FOREST CLASSIFIER

Model	Precision	Detection Accuracy
PARAFAC	0.9891	0.9396
PCA	0.9709	0.9121
PWO	0.9718	0.9396
PWO + Likelihood	0.9978	0.9725

TABLE III
FEATURE IMPORTANCE OF RANDOM FOREST

Residual Feature	Gini index
Difference up and down packets	0.5207
Up packets	0.1489
Down packets	0.1348
Difference up and down bytes	0.1150
Down bytes	0.0450
Up bytes	0.0356

as an alternative to PARAFAC for comparison purposes. As expected, for the five classifiers considered PARAFAC outperforms PCA as the latter loses structural information present in the data after converting tensors into matrices. The *Random Forest* classifier achieves better results for both methods. Moreover, preliminary evaluations indicate that models with two factors ($R = 2$) perform well, while increasing the number of factors does not improve the results. Therefore, the results in the sequel are obtained with *Random Forest* and with $R = 2$.

Next, we train the *Random Forest* classifier with the residuals obtained from set Tr_2 and evaluate the results using the residuals of set *Te*. Table II presents results for two evaluation metrics: Detection Accuracy and Precision. Detection Accuracy measures the percentage of anomalies detected, defined as $\frac{n_d}{n}$, where n_d is the number of detected attacks, and n is the total number of attacks in the set *Te*. We assume that an attack is detected if an anomaly is identified in at least one of the time slots that contain the traffic from that attack. Precision is calculated as follows: $\frac{\text{TP}}{\text{TP+FP}}$, where TP (True Positives) is the time (in minutes) where an attack occurs and is detected, while FP (False Positives) is defined as the time (in minutes) where an attack is wrongly detected. Therefore, Precision decreases when the number of False Positives increases. A model with a better detection rate (higher Detection Accuracy) is critical in scenarios where attacks have a major impact on the network. At the same time, a lower number of false positives (higher Precision) decreases the number of users incorrectly classified as attackers, reducing the chance of a user being wrongly affected by a countermeasure. For example, a legitimate customer might have its connection blocked if the classifier falsely reports an attack.

We compare the performance of PARAFAC and PCA models in the offline approach. Table II shows that PARAFAC achieves larger performance for both metrics. PARAFAC not only detects a higher percentage of attacks but also achieves higher precision, with a lower number of false positives. Moreover, although not reported, our results using PARAFAC show that all types of DDoS attacks evaluated have similar Detection Accuracy.

To evaluate the relevance of the six features retrieved from PARAFAC residuals we look at the Gini index-based importance metric from the *Random Forest* classifier, as shown in Table III. The most important residual feature is the *difference between upload and download packets* (Gini 0.5207) followed by *upload packets* (Gini 0.1489) and *download packets* (Gini 0.1348). We evaluate the classifier using only packet-rate based features and compare the results against those obtained with PARAFAC for all the features. The results show that the number of false positives is larger in the first case, with Precision decreasing to 0.9780 using only packet features compared against 0.9891 using all features.

Table II shows that PWO preserves the same Detection Accuracy for online decomposition in comparison to the offline method, while Precision decays from 0.9891 (PARAFAC) to 0.9718 (PWO). A small decrease in performance is not surprising taking into account that PWO is an online approach where the model is constantly updated as soon as a new data stream arrives. Nevertheless, PWO still achieves better results for both metrics in comparison to PCA. We also consider the time to detect an attack in the online scenario. The detection time of the PWO model is one minute for 86.55% of the detected attacks, while 99.41% of the attacks are detected within two minutes. A short detection time is essential to adopt fast countermeasures and mitigate the impact of an attack.

The histograms shown in Figure 3 suggest the use of gaussian mixtures (GMM) to model the residual distribution. To leverage this observation we fit two gaussian mixtures with two components and six dimensions, where each dimension is given by one of the features described in Section VI-C. The first GMM is trained using only minutes with attacks while the other mixture is trained using minutes without attacks. Then, for each time slice we compute the likelihood of each gaussian mixture for the observed residuals. Finally, those two likelihoods are taken as two additional features to the online PWO classifier, totaling eight input features. The results reported in the last line of Table II indicate that those two additional features can significantly increase precision and accuracy.

E. Spatio-temporal correlation

It is possible to further improve attack detection rates by correlating the classifier results for each home, since DDoS attacks are synchronized by nature. Mendonça et al. [12] propose a Bayesian decision problem using MAP criterion to detect synchronized attacks with high probability. Using the model parameterized with our results for the PWO online method ($|\mathcal{H}| = 812, P_D \approx 0.0014, p_{fp} \approx 2.64 \cdot 10^{-6}, p_{rc} \approx 0.8266, q = 0.05$) yields $m_0 \approx 4.21$. Therefore, the spatio-temporal correlation model assumes that a synchronized attack occurs if at least 5 users report an attack. It presents a great performance with the probability of false alarms (Type I error) equals $3.73 \cdot 10^{-16}$ and the probability of missing a synchronized attack (Type II error) equals $9.11 \cdot 10^{-11}$.

VII. APPLICATION II: DETECTING NETWORK DEGRADATION INTERVALS

We apply our methodology to the detection of degradation intervals in the ISP network. In the absence of reliable labels to identify anomalies and evaluate the results quantitatively, we rely on unsupervised clustering over residuals extracted by the offline method to group events with similar behavior.

An application example of the method is to automatically identify potential network problems affecting multiple users and to show the regions with poor performance.

A. Preprocessing

We use both latency and loss time series measured at one-minute intervals as the main performance metrics of interest. These metrics were collected on 2964 home routers between 19-August-2019 and 22-September-2019. Both metrics are obtained by sending a train of 100 ICMP packets at 10 millisecond intervals to a server located in the ISP network. Latency and loss measurements can be affected by home network user traffic (which we call cross-traffic) [21]. Therefore, we do not consider the value of minute samples when cross-traffic is greater than a threshold θ. Based on the users with the lowest nominal capacity in our dataset we set this threshold as $\theta = 2.5$ Mbps. After filtering out cross-traffic, we only consider time series with at least $\eta = 1000$ samples.

For some network failures, e.g. link failures, the communication between a client and the measurement server can be disrupted and no measurement sample recorded. Therefore, it is possible to infer periods of network unavailability from the lack of measurement samples, especially when multiple users do not simultaneously report measurement results. Hence, we encode each minute bin for which loss measurements are missing as having packet loss rate equal to 1. For analysis, we remove samples with cross-traffic above the threshold θ and periods when the measurement server is offline. This introduces missing samples that PARAFAC can easily handle. Finally, we use the log of each sample as input for the tensor decomposition. We considered 50282 multivariate time series with latency and loss values of one-minute granularity.

B. Tensor decomposition and residual extraction

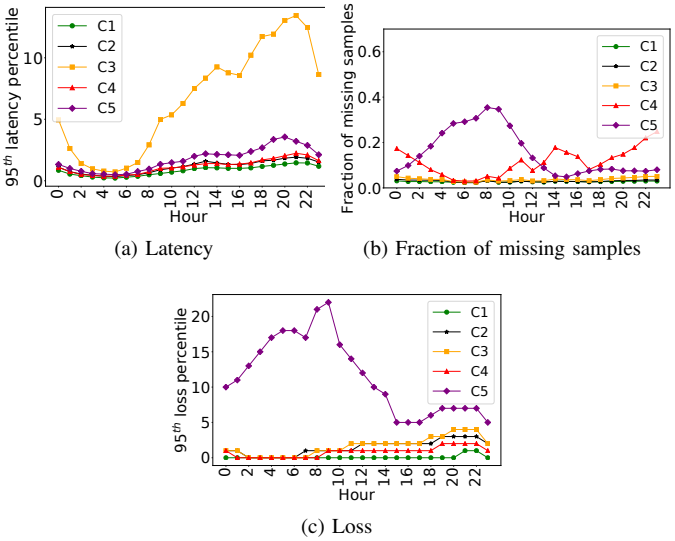
We model the measurement data as a tensor with three modes: (UD $\times$ network metric $\times$ minute). Since we use daily time series of latency/loss measurements as inputs, we get a third-order tensor $\mathcal{X} \in \mathbb{R}^{50282 \times 2 \times 1440}$. We use the *Split-Half Validation* method to set the number of factors $R = 4$.

We use metrics obtained from three different residual time series: latency residuals, loss residuals filtering samples with packet loss fraction equal to 1 and loss residuals without filtering. Each metric can be used to detect a different type of anomaly, such as network congestion and link or equipment failures. We extract three statistics for each residual time series: mean, standard deviation and 95^{th} percentile, totaling nine features.

We use metrics obtained from three different residual time series: latency residuals, loss residuals filtering samples with packet loss fraction equal to 1 and loss residuals without filtering

C. Anomaly clustering

1) Clustering results: We use K-Means for clustering the residuals due to its simplicity and interpretability. To select the number of clusters, we use the Elbow Method [4]. Five clusters are chosen. Before clustering the data we apply the z-score normalization to avoid that any feature dominates the analysis due to scaling.



(a) Latency (b) Fraction of missing samples

(c) Loss

Fig. 5. Summarized time series for each cluster

To investigate the meaning of each cluster we summarize all time series assigned to each cluster considering three metrics: latency, loss and amount of missing samples. To evaluate the latency per cluster we apply a normalization obtained by subtracting out the lowest value from each daily time series in order to infer packet queuing times during congestion periods.

Figures 5(a), 5(b) and 5(c) show summaries of each metric in all clusters. Users of cluster C1 experience good quality of service (low latency, low loss, low unavailability). Cluster C2 contains time series with moderate losses but low unavailability and low latency. Cluster C3 contains time series with high latency and moderate losses. The time series of cluster C4 are from users that experience high unavailability periods, while cluster C5 contains time series with both high unavailability and loss rates. Note that we cluster UD pairs, i.e., daily measurement time series of different users. Therefore, a user can be assigned to different clusters at different days, as we show in the next section.

2) Spatial correlation: To identify periods experiencing performance degradation affecting multiple geographically "close" users, we spatially correlate the clustering results and ISP topology information. The spatio-temporal correlation algorithm assumes that the routes between home-routers and the measurement server are static during each measurement interval. Consequently, the network topology can be represented by a tree structure at each measurement interval. We expect clients that share the same ISP network paths should exhibit similar performance inside the ISP network in terms of congestion and failures. We analyze the fraction of users assigned to each cluster at each day of the dataset.

We exemplify the results of the spatial correlation for a specific region of the network. Similar results are obtained for other network regions. Figure 6 shows the daily fraction of UD pairs per cluster. Usually the majority of users are associated with cluster C1 and few losses are observed. However a large number of users are associated with cluster C4 at day 10, when

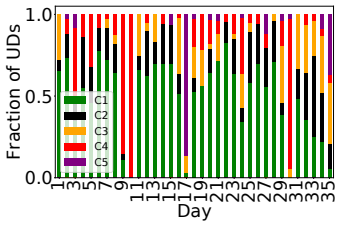
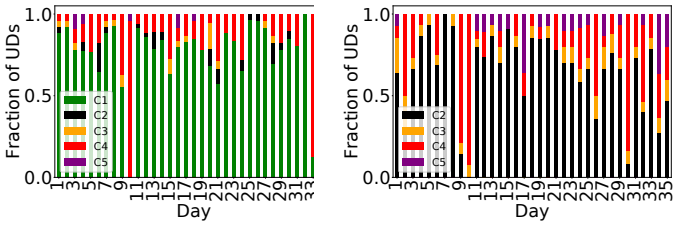


Fig. 6. Spatial correlation example

multiple time series have missing samples between 1 P.M. and 5 P.M. Another type of event detected by the spatial correlation occurs on day 17, when periods with missing samples between 6 A.M. and 8 A.M. and high losses between 7 P.M. and 9 P.M. were observed and several users are assigned to cluster C5.

The framework results suggests a measure of quality for each region of the network based on the number of UD pairs assigned to the cluster representing good performance (cluster C1). Figure 7 presents the clustering results in two different portions of the network. It can be seen that one region consistently presents a high fraction of users associated with better performance (Figure 7(a)), although performance degradation periods can be identified on days 10 and 33. At the same time, Figure 7(b) shows a region where no users are assigned to cluster C1.



(a) Region A (b) Region B

Fig. 7. Network performance obtained from residual clustering

VIII. Conclusion

In this work, we propose a method based on tensor decomposition to detect network anomalies. We apply the PARAFAC method and extract the residuals obtained by the model in order to detect abnormal behavior. We also propose a new online tensor decomposition method that efficiently extracts the normal subspace and detects anomalies with good performance. We show the flexibility of our method, using two different applications as examples. First, we consider DDoS attack detection using supervised techniques. The results show that we can obtain high values for Detection Accuracy and Precision using different classifiers. In addition, our method has better performance and robustness when compared to PCA. Then, we employ the proposed methodology with unsupervised techniques to identify periods within which network performance deteriorates and show how QoS problems can be detected on different parts of an ISP's topology.

Ackowledgments: This work was partially supported by grants from CNPq, CAPES, FAPERJ and by international cooperative grants from MCTIC-FAPESP, NSF_EAGER_1740895/MCTIC-RNP and Army Research Labs (ARL) No.W̃911NF-17-2-0196.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] S. Fadilpašić. (2019) AWS hit by DDoS attack. [Online]. Available: https://tinyurl.com/itpronews

[3] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Comp. Comm. Review*, vol. 34, no. 4, 2004, pp. 219–230.

[4] ——, "Mining anomalies using traffic feature distributions," in *ACM computer communication review*, vol. 35, no. 4, 2005, pp. 217–228.

[5] R. Bro, "Parafac. tutorial and applications," *Chemometrics and intelligent laboratory systems*, vol. 38, no. 2, pp. 149–171, 1997.

[6] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 3, p. 11, 2008.

[7] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Trans. Net. and Service Manag.*, vol. 16, no. 3, pp. 800–813, 2019.

[8] K. Maruhashi, F. Guo, and C. Faloutsos, "Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis," in *Advances in Social Networks Analysis & Mining*, 2011, pp. 203–210.

[9] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Graph based tensor recovery for accurate internet anomaly detection," in *IEEE INFOCOM 2018*, 2018, pp. 1502–1510.

[10] F. Silveira, C. Diot, N. Taft, and R. Govindan, "Astute: Detecting a different class of traffic anomalies," *ACM SIGCOMM CCR*, vol. 41, no. 4, pp. 267–278, 2011.

[11] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer IoT devices," *IEEE Security and Privacy Workshops*, pp. 29–35, 2018.

[12] G. Mendonça, G. H. A. Santos, E. d. S. e Silva, R. M. Leão, D. S. Menasché, and D. Towsley, "An extremely lightweight approach for ddos detection at home gateways," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5012–5021.

[13] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[14] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 636–650, 2016.

[15] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the parafac decomposition of a third-order tensor," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2299–2310, 2009.

[16] S. Zhou, N. X. Vinh, J. Bailey, Y. Jia, and I. Davidson, "Accelerating online cp decompositions for higher order tensors," in *ACM SIGKDD Knowledge Discovery and Data Mining*, 2016, pp. 1375–1384.

[17] R. A. Harshman, ""How can i know if it's real?" a catalogue of diagnostics for use with three-mode factor analysis," *Research methods for multimode data analysis*, pp. 566–591, 1984.

[18] U. Lorenzo-Seva and J. M. Ten Berge, "Tucker's congruence coefficient as a meaningful index of factor similarity," *Methodology*, vol. 2, no. 2, pp. 57–64, 2006.

[19] E. Auchard, "German Internet outage was failed botnet attempt: report," https://tinyurl.com/reutersoutage, Reuters.

[20] N. Blenn, V. Ghiëtte, and C. Doerr, "Quantifying the spectrum of denial-of-service attacks through Internet backscatter," in *Conference on Availability, Reliability and Security*, 2017, p. 21.

[21] S. Sundaresan, W. de Donato, N.Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: A view from the gateway," in *ACM SIGCOMM 2011*, 2011.