Leveraging Website Popularity Differences to Identify Performance Anomalies

Giulio Grassi, Renata Teixeira, Chadi Barakat, Mark Crovella

Leveraging Website Popularity Differences to Identify Performance Anomalies

Giulio Grassi
Inria
Paris, France
giulio.grassi@inria.fr

Renata Teixeira
Inria
Paris, France
renata.teixeira@inria.fr

Chadi Barakat
Université Côte d'Azur, Inria
Sophia Antipolis, France
chadi.barakat@inria.fr

Mark Crovella
Boston University
Boston, USA
crovella@bu.edu

ABSTRACT

Web performance anomalies (*e.g.* time periods when metrics like page load time are abnormally high) have significant impact on user experience and revenues of web service providers. Existing methods to automatically detect web performance anomalies focus on popular websites (*e.g.* with tens of thousands of visits per minute). Across a wider diversity of websites, however, the number of visits per hour varies enormously, and some sites will only have few visits per hour. Low rates of visits create measurement gaps and noise that prevent the use of existing methods. This paper develops WMF, a web performance anomaly detection method applicable across a range of websites with highly variable measurement volume. To demonstrate our method, we leverage data from a website monitoring company, which allows us to leverage cross-site measurements. WMF uses matrix factorization to mine patterns that emerge from a subset of the websites to "fill in" missing data on other websites. Our validation using both a controlled website and synthetic anomalies shows that WMF's F1-score is more than double that of the state-of-the-art method. We then apply WMF to three months of web performance measurements to shed light on performance anomalies across a variety of 125 small to medium websites.

1 INTRODUCTION

As more and more of our society and daily lives migrate online (from shopping to entertainment and work to mostly everything under the current pandemic), web performance becomes ever more crucial. Slow web load times can have significant impact on the revenues of online businesses, as users give up on browsing due to temporary performance degradations [1]. For example, Google reports that "increasing web search latency from 100 ms to 400 ms reduces the daily number of searches per user by 0.2% to 0.6%" [2] and a one second delay in page load time would cost Amazon an estimate of $1.6 billion in annual sales [3]. As such, website administrators spend significant effort to optimize the performance of their webpages—a task that requires accurate and efficient methods to detect and diagnose poor web performance.

Web performance monitoring and diagnosis have thus received considerable attention from research and industry. A number of efforts have focused on designing metrics to capture webpage performance (for example, Page Load Time, Above-The-Fold-Time [4], or SpeedIndex [5]) and understanding how these metrics relate to the user quality of experience [6–8]. Webpages are often instrumented to passively collect at least some of these metrics. In some cases, websites contract third-party companies such as Dynatrace [9], Pingdom [10], IP-Label [11], or Akamai [12] to monitor the performance of their websites on their behalf. An important task for such services is detecting *web performance anomalies* — abnormally high values of, for instance, page load time (PLT).

Unfortunately, however, both website administrators and third-party monitoring services lack effective tools to automatically detect the rare occurrences of web performance anomalies within the many visits to a website. To date, methods for web performance anomaly detection have focused on search response times [13, 14]. Such methods are designed for very popular websites (*e.g.* for Bing and Baidu, with tens to hundreds of thousands of visits per minute). However, when using passive measurement for more general-purpose websites, a significant challenge arises due to the Zipfian distribution of overall popularity. Across a collection of websites, usually a small number of sites will account for most visits and a vast majority of sites will be less popular [15]. This problem is compounded by the fact that measurement of site performance is expensive, and in many cases only a small sample of visits to a site can be instrumented and measured.

This highly variable quantity of performance measurements

available across websites causes severe problems for traditional anomaly detection methods. First, anomaly detection for general purpose websites must deal with *measurement gaps* – time periods with no measurements. These gaps can happen because when a less-popular website takes sampled measurements, there can be periods where none of the visits are sampled. Second, low numbers of measurements increase the effect of *measurement noise*. For example, during a certain period, page load time may increase simply because the lower number of measurements happen to include many users who have a poor cellular connection.

On the other hand, the performance of websites is often measured by third parties. In this setting we observe that there is an opportunity to overcome challenges due to variable data volume by looking *across* websites. That is, we propose to "borrow from strength" using measurements of many websites simultaneously to overcome the challenges of spare per-site data. In this paper, we use this insight to develop a web performance anomaly detection method that overcomes the challenges of highly variable measurement volume. To demonstrate our method, we leverage data collected by a web performance monitoring company (presented in Sec. 3). Our method can learn patterns from a subset of websites to help "fill in" the missing data on other websites. The method we develop in Sec. 4, called WMF for *Web Matrix Factorization*, leverages matrix factorization and the dataset's low dimensionality to identify both normal web performance and performance anomalies. We validate WMF in a controlled live setting where we are able to induce network delay to a server, as well as by injecting synthetic anomalies in the dataset (Sec. 5). Our results discussed in Sec. 6 show that WMF achieves a recall of about 0.9 and a precision of about 0.98 in controlled experiments. Precision and recall are lower in the evaluation with synthetic anomalies, because the dataset contains anomalies that we have not introduced, but WMF's F1 score is more than double that of state-of-the-art time series methods for anomaly detection of web performance [13]. This result confirms that it is non-trivial to adapt existing methods developed for popular websites to work on less popular sites.

We then apply WMF to detect anomalies on three months of web performance measurements across 125 websites (Sec. 7). This analysis sheds light on web performance anomalies in a heterogeneous set of small to medium websites. Only 56 out of the 125 websites experience any web performance anomaly during our study. Most anomalies are short (lasting less than one hour), but we detect a few events that last for more than ten hours. We also show that some websites had at least one anomaly every five days on average. We conclude that an automated web anomaly detection tool such as WMF can assist web administrators or third-party monitoring companies to quickly detect anomalous events despite the challenges posed by heterogeneous measurement volume across websites.

2 BACKGROUND

To place our work in context, we review in this section common web performance metrics and existing methods for web performance anomaly detection.

2.1 Web performance metrics

Modern methods for measuring web page performance typically compute the time elapsed between events occurring during the loading of the page. Figure 1 presents an overview of relevant events. Assuming direct access to the page (*i.e.* no redirect) the browser first issues a DNS request to discover the IP address of the server, and then establishes a TCP connection to perform an HTTP request. The browser loads the Document Object Model (DOM), which represents the page as nodes and objects, and then progressively renders the document. Finally, once all objects have been downloaded and rendered, including all the dependent resources such as stylesheets and images, the page is complete.

The most common web performance metric is the Page Load Time (PLT), which measures the time between the beginning of navigation and the `loadEventEnd`. PLT indicates how long it takes to fully load a page after the user clicks on it. Other metrics break down PLT. For example, Time To Interactive (TTI) indicates the point when the user can interact with the page once it has been rendered; Time to First Byte when the first byte of the response is received; and Time to First Paint, the time at which the first object is painted by the browser. Although PLT is the most popular web performance metric, the literature is rife with metrics that aim to better capture the time the user considers a page loaded (*e.g.* Above the fold time [4], User-Perceived Page Load Time [16], First Contentful Paint [17] and Largest Contentful Paint [18]) as well as metrics that capture the overall experience of loading the webpage instead (*e.g.* Speed Index [5], Perceptual Speed Index [7]). The methods we develop in this paper can in principle be applied to any of these metrics. For concreteness, we use the set of metrics defined in Sec. 3.1.

2.2 Anomaly detection: Existing methods

In this paper, we are concerned with *web performance anomalies,* by which we mean abnormally poor (*i.e.* high) values of one of the web performance metrics (for instance, PLT). Because web and Internet measurements in general are noisy, a single outlier measurement is usually not significant enough to a website operator to merit a response. Hence, in this paper we focus on anomalies that last tens of minutes and that affect a considerable fraction of users.

A number of techniques have been proposed in the literature for anomaly detection of web measurements, and more

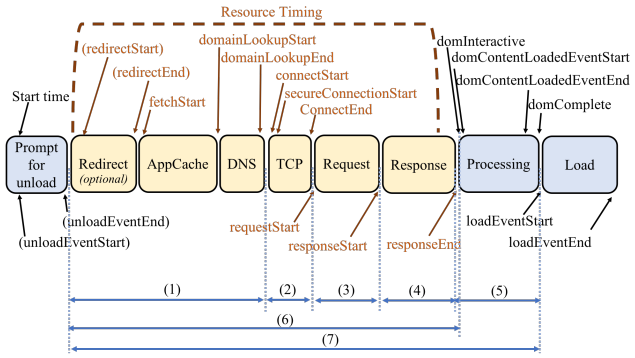


Figure 1— What happens when the user opens a webpage. Web performance Metrics: (1) DNS Response Time; (2) TCP Connection Time; (3) First Byte Received; (4) Object Response Time; (5) Page Processing Time; (6) Time To Interactive; (7) DOM Completion Time.

generally, network measurements. We divide existing methods into *temporal* and *spatio-temporal* approaches.

Temporal methods. The most common way to study temporal data is to build a time series model of measurements in aggregated form (*e.g.* average or median of measurements in a time bin). Deviations from model predictions can then be used to identify anomalous time points. In this category we find Seasonal-Trend Decomposition using Loess (STL) [19] methods, which split the time series into three signals: the trend, the seasonal, and the residual components. The latter is normally analyzed to detect anomalies. In the domain of web performance measurements, the Week-over-Week (WoW) method relies on STL to detect anomalies on Search Response Time measurements performed on a major web search provider [13]. Other temporal approaches for anomaly detection are based on Autoregressive Integrated Moving Average (ARIMA) [20–22] and Exponential Smoothing [23] (with Holt-Winters seasonal method [24] being the most common), where historical values are used to forecast the next value of a time series, and anomaly detection is built on top of values far from the outcome of the model. Finally, neural networks methods such as LSTM Autoencoder [25, 26] can be used to detect anomalies, where the model is trained with a set of 'normal' data and then tries to reproduce the rest of the dataset. A degradation in the accuracy of the reconstructed signal means an anomaly is present.

A key concern when using time series analysis methods is the need for reliable values for every time bin. Whenever a time bin has no data one generally must interpolate the missing values. Errors in this estimation of the missing values will compromise the accuracy of the anomaly detection process on the known values. In our setting where many websites have low measurement volumes and can have many missing values, this becomes a critical issue.

Spatio-temporal methods. Spatio-temporal methods are commonly adopted in case of high dimensional datasets with timeseries measurements performed on different entities or at different locations. These techniques not only analyze the temporal relation of the signals in the datasets, but they also exploit the correlation among the different signals. A popular technique is Principal Component Analysis (PCA), which performs a linear mapping of the data to a lower dimensional space so as to represent the data using fewer components that describe the majority of the variance of the entire dataset. The low-rank representation of the dataset typically captures the 'normal' behavior, and deviations (residuals) from the low-rank representation allow identification of anomalies. PCA has been broadly used for anomaly detection, for instance in wide-area communication networks [27, 28], datacenter services [29], and social network activity [30]. A generalization of PCA to tensor factorization has been also adopted for anomaly detection [31].

However, as a data analysis technique, PCA has a limitation similar to time-series methods in that it cannot be used directly on data with missing values. An alternative to PCA that can build a model of data with missing values is *low rank matrix factorization,* also termed *matrix completion* [32–34]. A number of efficient algorithms have been developed for this problem [35–37]. Like PCA, matrix factorization seeks to find linear relations among the different signals in order to describe the majority of the data with a low rank matrix. However, it is designed to operate on matrices with missing data, and importantly, it can construct predictions of the missing data based on the low-rank approximation.

As noted above, a central challenge of working with website data is that there is an enormous variation in measurement volume across sites, as well as time periods with no sampled visits. This leads to many cases of missing data in our measurement matrices. It is thus very hard to build either time series or PCA models on such data (and such methods perform poorly, as we show below). Hence in this paper we adopt low-rank matrix factorization and adapt it to the problem of anomaly detection (described in detail in Section 4). While this general methodology has appeared in one prior work [38], to our knowledge this is the first work to apply it to anomaly detection problems in networking, and the first to recognize its power for addressing the data challenges presented by highly-variable website popularity.

3 DATASET

This section describes the dataset we study in this paper. We highlight how the small to medium popularity of the websites in this dataset results in a sparse dataset that is particularly challenging for typical anomaly detection methods.

3.1 Overview

The primary dataset we use in this paper is collected by ip-label, a company that sells both passive and active website performance monitoring solutions.

Web performance metrics. Ip-label embeds Javascript code in the webpages of its customers to measure web performance via the Navigation Timing API. Measurements are collected within the user's browser, then exported to a database maintained by the company. Website operators pay the monitoring service based on both the number of requests monitored and the number of metrics logged. To reduce costs, website operators often select a subset of webpages and visits to monitor, and collect only a few of the metrics available from the Navigation Timing API.

Our study relies on the dataset collected on 125 websites from July 29 to October 27, 2019 (mostly websites of private companies with few e-commerce and public institutions). Users access these sites from 235 countries across all continents, but 78% of the accesses come from Europe and 18% from Asia. We use the term *visit* to refer to a user accessing a webpage (or URL) within a website. The passive monitoring solution logs web performance of visits to a monitored website together with meta-data about the client's device.

Table 1 presents the available web performance metrics in the dataset together with the number of websites and visits that collect each metric. Metrics are defined based on the events shown in Figure 1. In some cases our metrics are slightly different from standard metrics in the literature. For example, the dataset does not contain the standard definition of Page Load Time (which is `loadEventEnd` minus `navigationStart`), because the company customizes its measurements based on the needs of individual customers. Instead, it captures a *Customized Page Load Time*, which starts at `responseStart` and ends at some point during the load of the page where the page is considered loaded by the page designer (so it will trigger some time between `loadEventStart` and `loadEventEnd`). *Time To Interactive* measures the time for the browser to parse the page and set the page readiness as interactive (it is the time of the `domInteractive` event minus that of the `unloadEventStart`). *DOM Completion Time* indicates the overall time between `unloadEventEnd` and `loadEventStart`. *Object Response Time* measures the time for the response (i.e., the interval between `responseStart` and `responseEnd`). Refer to the caption of Figure 1 for the definitions of the remaining metrics in Table 1.

Note that if the page completely fails to load, then we have no measurement. In contrast, partially loading a page may still trigger some of the measurements, depending on the metric *e.g.* a First Byte Received measurement does not require the full page to be loaded. While our method is designed to work with such data, the absence of measurement by definition means we cannot detect any anomaly at that timepoint. Hence a complete outage of a website is not an anomaly that our system is designed to detect, so we exclude outages from the list of anomalies we target.

Metric	#Websites	#Visits
Customized Page Load Time	94	233M
Time To Interactive	51	166M
DOM Completion Time	45	79M
Object Response Time	43	75M
Page Processing Time	43	74M
First Byte Received	43	63M
Connection Time	42	23M
DNS Response Time	39	16M

Table 1— Web performance metrics in the dataset. *#Websites* indicates how many websites report measurements for the given metric, while *#Visits* reports the number of visits that trigger a measurement for a given metric.

Meta-data. For each visit, the script also logs meta-data about the client. These are listed in Table 2: IP address, IP-based geo-location, client's ISP and time of the visit, as well as information extracted from the user-agent—as browser type, client's device type (*i.e.* mobile or PC) and brand. In some cases, part of the meta-data may be missing (for instance, when the geo-location service is unable to estimate the location of a device). Table 2 also reports the percentage of visits that have missing meta-data fields.

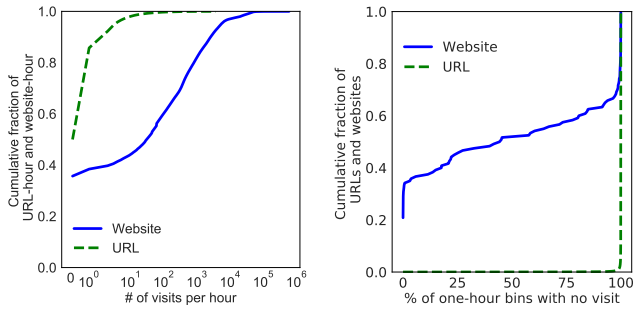
3.2 Challenge: Data sparsity

As described in Sec. 2, prior methods for web anomaly detection were developed and evaluated for highly-popular websites such as Bing [13] (~270k visits per minute worldwide [39]) and Baidu [14] (~10k visits per minute). The websites in the dataset presented in the previous section, however, are considerably less popular. Fig. 2(a) shows the cumulative distribution function of the number of visits per URL and per website in one-hour bins. We see that a vast majority of URLs have very few visits per one-hour bin: over 50% of bins have no visits, and less than 20% have more than one visit in a time bin. We can improve the per-bin counts by aggregating visits to all monitored URLs within the same website, which helps somewhat, reducing the fraction of bins with no visits to less than 40%. Unfortunately however, page structure and content can be drastically different across the URLs of a given website, which makes it difficult to interpret the values of web performance metrics. As a result, the average or median TTI of a website can, for instance, vary significantly between time

Attribute	Cardinality	Missing meta-data[%]
Website	125	0%
URL	8876k	0.06%
User's ISP	16k	1.3%
User's AS	17k	1.3%
User Agent	4.8M	0.02%
OS name / version	61 / 1340	0.2% / 11.5%
Browser Name / version	50 / 1630	0.3% / 14.1%
JS version	10	0%
Device type / brand	9 / 56	0.2% / 9.6%
Country	235	0%
Region	392	14.2%
Postal code	41k	84%
City	51k	17%

Table 2— Meta-data logged per visit (for the period July 29, 2019 - October 27, 2019, corresponding to 290 million visits). The percentage of missing meta-data refers to the ratio of the number of visits with missing meta-data to the total number of visits.

bins simply because the subset of accessed URLs in a bin varies over time. And nonetheless, even when we aggregate all visits to different URLs of the same website, Fig. 2(b) shows that for almost 50% of websites, at least half of their one-hour bins have zero visits.



(a)— Number of measurements per hour.

(b)— Percentage of one-hour bins with no measurements.

Figure 2— Aggregation by time (hourly representation) and space (by URL or by website) of visits triggering web performance measurements.

The nature of this data illustrates the challenges of anomaly detection across general-purpose websites. First, even when aggregating measurements in fairly large, one-hour time bins, a significant fraction of bins still have no data. Second, the number of measurements per hour is often small, which makes the statistics less stable. For example, the average PLT may vary significantly between time bins because users visit from poorer connections or devices. We note that previous work [13] has actually filtered out users accessing from mobile devices to eliminate some variation in metrics, but for our problem such an approach would reduce even further the number of data points for analysis.

3.3 Data representation choices.

The challenges presented in the previous subsection drive our data representation choices, which we describe here.

Aggregating samples of multiple URLs of a website. As just discussed, Fig. 2 illustrates the challenges of using per-URL measurements for anomaly detection. Hence, to reduce the fraction of empty bins, we aggregate visits to different webpages of the same website. The content and structure across these pages, however, can be considerably different, which leads to very different baseline web performance. To address this problem, we adopt a strategy of normalizing data at the per-URL level. In fact, we normalize measurements for each web performance metric to zero-mean and unit-variance within each URL. This normalization removes differences caused by varying page content and yet ensures that anomalies still appear as outliers. Thus each measurement, x, of the performance of a webpage, u, is transformed to its normalized value as $x' = \frac{(x-\mu_u)}{\sigma_u}$, where μ_u is the mean of the collected measurements on page u and σ_u the standard deviation of these measurements. This allows us to more meaningfully mix performance measurements of a given metric across different pages of a website.

One-hour time bins. In addition to aggregating measurements across webpages, we also aggregate measurements over time. We divide time into fixed-size bins and summarize all the samples that fall into a bin. We evaluate the number of samples in a bin to help select the appropriate bin size. Fig. 3 presents a box plot (with median, 25th and 75th percentile) of the number of measurements per bin as we vary the bin size. With 15-minute bins, only 25% of the bins have more than 100 visits, and more than half of the bins have less than 15 visits. When the number of visits in a bin is too small, the aggregate metric may be biased by confounding factors (*e.g.* a small set of poorly connected devices visiting the website). On the other hand, increasing the size of the time bins too much will tend to hide anomalies that occur with shorter time duration. Based on our analysis, we select a one-hour bin as good trade-off. Note that when applying our method to other datasets, it may be desirable to select larger or smaller bin sizes based on the traffic load of those datasets (*e.g.* smaller when working with more popular websites overall).

Summarizing samples in a bin. Average is commonly used to summarize samples. Nevertheless, due to the intrinsic elevate noise levels caused by the limited volume of measurements, we opt for percentiles, because of their robustness. We

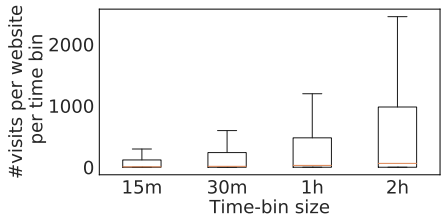


Figure 3— Number of per website visits with different time bin sizes (15 and 30 minutes, one and two hours).

evaluate the effect of using the average, median as well as the 25th, 75th, 95th percentiles on the accuracy of anomaly detection. In contrast with the median or the 25th percentile, the 75th percentile better captures fluctuations of web performance even when only a small subset of samples is anomalous. At the same time, it is less sensitive compared to average and 95th percentile to extremely large web performance values of few samples, which may be caused by few poorly connected devices. Our results (presented in Appendix A) show that using the 75th percentile achieves the highest F1 score in general. We thus adopt the 75th percentile for its better visibility on the different types of anomalies, while recognizing that other metrics may be helpful in other contexts.

Multiple web performance metrics. A final strategy we use to maximize the amount of data available for anomaly detection is to combine samples of multiple web performance metrics within the same matrix. Had we only considered a single metric, we would be discarding a significant fraction of available data. For example, even Customized Page Load Time, which is the most popular metric in the dataset (Table 1), is measured in only approximately 80% of visits. In addition, different metrics help capture different types of anomalies. For example, First Byte Received is more sensitive to high delays, whereas TTI and Customized Page Load Time are also going to increase under losses. This leads to one time series per web performance metric collected for each of the monitored website. Given that we normalize values of individual metrics so that distributions per metric have zero-mean and unit-variance, it is therefore reasonable to analyze time series of multiple metrics together in the same matrix.

Notation. Summing up, for each website and performance metric pair, we create a time series of the hourly 75th percentile of measurements. We obtain 425 time series, which we represent into an $n \times m$ matrix, $\mathbb{M}$. A row, i, corresponds to a time bin and a column, j, to a (website, performance metric) pair. The element (i, j) is the 75th percentile of the measurements of the (website, performance metric) pair, j, during the time bin, i. The dimensions of $\mathbb{M}$ depend on the dataset. In our case, $m = 425$ and $n = 2184$, which corresponds to three months of measurements. In a long running system, we expect to select n time bins in a sliding window fashion. Note that for the methods we discuss in the next section to work, we need to consider at least a few months of data to capture any weekly patterns in the dataset.

4 METHOD

Given web performance of multiple websites over time represented in $\mathbb{M}$, our goal is to identify web performance anomalies – cells of $\mathbb{M}$ (*i.e.* a specific web performance metric for a specific website at a specific time) with abnormally high values. Note that the one-hour bins we use to aggregate web performance samples ensure that we focus on significant anomalies that affect a large fraction of visits during the hour.

Applying standard anomaly detection methods to $\mathbb{M}$ is challenging given the large number of empty cells (46% of the cells in $\mathbb{M}$ are empty). As described in Sec. 2, matrix factorization is capable of constructing a low-rank matrix approximation even when there are many missing values. If a matrix has a low-rank approximation, $\mathbb{M}_L$, we can use this approximation to describe the 'normal' component of the data. The difference between the original matrix and the low-rank approximation represents the residual component, $\mathbb{M}_R$, which contains the anomalous part of the data. This section first evaluates our assumptions, then it presents our method, which we call WMF for Web Matrix Factorization, to compute $\mathbb{M}_L$ and to detect anomalies using $\mathbb{M}_R$.

4.1 Assumptions

Our method is fundamentally spatio-temporal, meaning that we identify the normal website performance from patterns found in the entire $\mathbb{M}$. The intuition is that factors such as the daily and weekly pattern of user access, common cloud providers, or shared network paths create partial correlation across websites. If we find these inter-dependencies (despite the large amount of missing data) we can use measurements of other websites to evaluate whether at a particular time bin the performance measurements of a given website are anomalous or not. This approach requires that in a given time when one website has no visits (*i.e.* a cell in $\mathbb{M}$ is empty) other websites have visits, and that there exist linear dependencies between the performance of different websites (*i.e.* that we can approximate $\mathbb{M}$ with a low rank representation). We evaluate these two assumptions before presenting our method to estimate the normal form of $\mathbb{M}$.

Websites can learn from one another. In every time bin in the dataset, there are at least 136 out of 425 website-metric pairs with at least one visit. At the same time, each row contains at least 125 empty cells (Fig. 4). Our method uses the fact that websites with no measurements at a given time can potentially rely on the data from other websites with visits to help reconstruct their 'normal' signal.

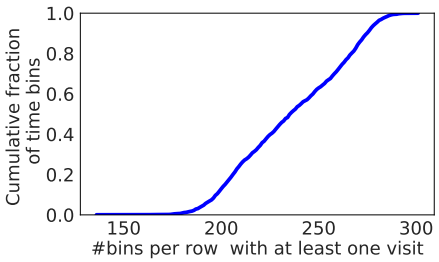


Figure 4— Number of cells in $\mathbb{M}$ with at least one visit per one-hour time bin.

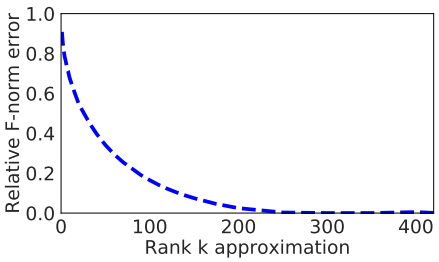


Figure 5— Rank reduction: relative F-norm error of rank k approximations of $\mathbb{M}$.

Low-rank representation of $\mathbb{M}$. The rank of a matrix indicates the number of linearly independent columns of the matrix. If $\mathbb{M}$ can be approximated with a matrix of rank lower than the number of columns, it means that there are linear dependencies among some of the columns of $\mathbb{M}$. This approximation allows to separate the normal pattern (captured in the low-rank approximation) from the anomalous pattern.

For a partially observed matrix A, we denote the set of known values in A as Ω. Given a matrix A in which only the entries Ω are observed, we denote the error of using B to approximate A as

$$\|(A-B)_\Omega\|_F^2 = \sum_{(i,j)\in\Omega} (a_{ij}-b_{ij})^2.$$

To approximate an $n \times m$ matrix A via a low rank matrix, we seek to find factors U and V, such that U is $n \times k$, V is $m \times k$, and the rank k approximation of A is UV^T. This leads to the following minimization objective:

$$\min_{U,V} \left\| \left(\mathbb{M} - UV^T\right)_\Omega \right\|_F^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2\right), \qquad (1)$$

where Ω is the set of observable entries, U and V are the factor matrices and λ is a regularization parameter for U and V to avoid overfitting (with $\|X\|_F^2 = \sum_{ij} x_{ij}^2$).

To solve Eq. (1) for a given k and λ, we use an algorithm based on alternating least squares (ALS) [37]. The algorithm alternates between finding the best U and V in order to achieve the minimization objective. In practice, the ALS algorithm is fast and robust; it is known to typically converge quickly to a good rank k matrix UV^T for approximating $\mathbb{M}$.

To illustrate that a low-rank approximation is appropriate for our data, we plot in Fig. 5 the relative error of each rank k estimate of $\mathbb{M}$ (*i.e.* $\|(\mathbb{M} - UV^T)_\Omega\|_F^2 / \|\mathbb{M}_\Omega\|_F^2$). We see that the error rapidly decreases with k and reaches zero at approximately $k = 250$. This figure shows that we can indeed approximate the full 425-column matrix $\mathbb{M}$ with only 20% error using a matrix of rank between 70 and 100 (and that the 125 websites are not completely independent).

4.2 Estimation of the normal component

Different techniques exist in the literature for the estimation of the normal low rank component of a matrix. We evaluate ALS [37], SVD [40], PCA and LMaFit [35] and provide comparison results in Appendix B to estimate, $\mathbb{M}_L$, the low-rank representation of $\mathbb{M}$. We observe that ALS achieves the highest F1 scores; we hence apply ALS in WMF. We set $\lambda = 0.25$ and $k = 70$ because these values achieve the maximum F1 scores for anomaly detection as described in Appendix C. Note that these settings are not chosen to find the smallest error when representing $\mathbb{M}$ (as shown in Fig. 5), but rather to provide the best representation of the normal form of $\mathbb{M}$ for anomaly detection.

4.3 Anomaly detection

We next estimate the anomalous component, $\mathbb{M}_R$. Given the original matrix $\mathbb{M}$ with the per-hour average of all the web performance metrics for each website, and its low-rank approximation, $\mathbb{M}_L$, we compute the difference between $\mathbb{M}$ and $\mathbb{M}_L$. Note that $\mathbb{M}_L$ has no missing values thanks to the ALS method that fills in the missing values in such a way to preserve the structure of the low-rank approximation. When a cell in $\mathbb{M}$ is empty, however, we can safely ignore it while calculating the difference, because no users visited the site at that time, so if there was an anomaly it would have no impact. The resulting matrix $\mathbb{M}_R$ indicates the extent of the anomalous component for each website-metric at every one-hour time bin. We then use $\mathbb{M}_R$ to detect anomalies.

Anomalies on a website appear in a single column of $\mathbb{M}$, or at most in few of them if the website collects measurements for other metrics and the same inefficiency impacts more than one metric. To detect such anomalies, we identify "large" residual values in $\mathbb{M}_R$. Anomalies in $\mathbb{M}$ are by definition far from the normal values and thus should not appear in $\mathbb{M}_L$, but only in the residual component matrix, $\mathbb{M}_R$. We label cells in $\mathbb{M}_R$ as anomalous if they are larger than a threshold, τ. Sec. 5.2 evaluates the accuracy of our method varying τ. We find that τ = 99.9th percentile of the values in $\mathbb{M}_R$ achieves the highest F1 score for our dataset.

4.4 Running time

We report here the running time of the analysis and break it down at each step. With an Intel(R) Xeon(R) Gold 6150 with 8 cores at 2.70GHz and a 2184 x 425 matrix (corresponding to three months of data), computing $\mathbb{M}_L$ and $\mathbb{M}_R$ takes on average 2 minutes and 26 seconds. The second step of the analysis, detecting the anomaly given the residual matrix, requires instead on average 0.6 seconds. The entire analysis is implemented in Python. An interesting avenue of future work is to optimize the implementation for real-time anomaly detection.

5 VALIDATION SETUP

The evaluation of any anomaly detection method is challenging due to the lack of ground truth, *i.e.* time bins that contain known anomalies and others that are known not to contain any anomalies. We rely on two approaches to validate WMF. First, we conduct experiments where we deliberately introduce network delay anomalies on the path to a server we control. Second, we introduce "synthetic anomalies", where we artificially increase the values of some of the entries in $\mathbb{M}$. The rest of this section describes our validation method, evaluation metrics, and the baseline method we compare against in the next section.

5.1 Controlled website

Monitored website. We host a website on a server in our lab instrumented with the same web monitoring software used to collect the dataset described in Sec. 3.1. We instrument five of the pages of the website. Three of the webpages have only static content, the first with only a short text and one image, the second with longer text and several small pictures spread across the entire page, and the third with a larger size image. The other two pages have, in addition to static images, content retrieved by Javascript code from other web services (*e.g.* Google maps, weather forecast widget, calendar widget, etc). Normally the website experiences very light load, with on average less than 2-3 visits per hour.

Injected anomalies. We introduce network delay and losses on the server hosting the website to later verify whether our method can detect the time bins with these issues as anomalies. We use Linux Traffic Control, *tc*, to periodically increase the delay of the outgoing traffic by either 500 ms or 1 s. Each delay setting requires approximately one month of experimentation time. We pick two values of delay increase in a range of values previously found to affect user experience [2, 41]. We similarly consider only two settings of losses, 10% and 30% (our preliminary experiments show that lower loss rates have negligible effect on web performance). Even at 30% loss rates the impact on web performance is minimal. Hence, we omit the results based on losses and focus in the rest of the paper on path delay anomalies. We introduce each network issue for 30 minutes, on average every 6 hours (with a Poisson process determining the starting time of the anomalous period). We select 30 minutes to challenge our method as issues that last more than one hour (which is the size of the time bin we consider) should be easier to detect as they affect all samples in a bin. At the same time, our goal is to detect anomalies that last for sufficient time for human intervention, so 30 minutes is already relatively short for challenging our method. We repeat experiments with each type of injected anomaly one-hundred times, from July 29 to November 15, 2019 (25 days for each type of injected anomaly).

Visits. Normally, our website has a small number of visits over time. To ensure that we have measurements during injected anomalies, we use two laptops (Apple and Linux machines) connected to two distinct WiFi networks (one residential and one lab WiFi) to visit the website every five minutes on average during the four-month experiment. We randomly pick one of the five instrumented webpages to visit. These automated visits bring the total number of visits to the website to ~30 per hour, maintaining intact the low popularity characteristics of the website. To reduce the impact of browser caching, we use incognito-mode navigation and at every visit we restart the browser.

5.2 Synthetic anomalies

We introduce synthetic anomalies in $\mathbb{M}$ to evaluate the method in a larger variety of settings. Our goal is to introduce realistic synthetic anomalies, where a single underlying issue (in our case, network delay) affects the web performance of multiple visits. We rely on the controlled website discussed in the previous section to build a regression model that captures the relationship between increased network delays and the degradation in web performance. We then apply this model to introduce web performance anomalies in $\mathbb{M}$.

Regression model: network delay versus web performance. We use the controlled website to collect samples of increased network delay and web performance (in particular, we measure TTI). We use the exact same setup as in Sec. 5.1 except that we now increase the delay in increments of 100 ms from 0 to 1 s and we inject each extra delay for a period of one hour. We then use linear regression to model the relationship between increased network delay and TTI. We build one model per webpage as the content and the structure of the pages differ. We then use these models to define a function, f, that takes as input a web performance sample, t, and a desired increase in network delay, d, and outputs the corresponding anomalous web performance value, $\hat{t}$. We present the regression models and define f in Appendix D. To introduce an anomaly, we then apply f with the desired d across all the web performance samples that correspond to

visits to a particular website. Finally, we use these anomalous web performance samples to recompute $\mathbb{M}$ as input for the method presented in Sec. 4 (this process includes re-running the per-url normalization process).

Injected anomalies. To introduce anomalies, we select a website and time bin and then increase the web performance of all samples in the corresponding cell of $\mathbb{M}$. We then re-run the per-url normalization process and recompute $\mathbb{M}$ (as described in Sec. 4). In particular, we evaluate with additional network delay, d, equal to 500ms, 1s, and 2s. For each d, we randomly pick 0.1% of cells in $\mathbb{M}$ among those with measurements to inject anomalies. To test the sensitivity of WMF to anomaly duration, we also inject anomalies with different durations.

5.3 Evaluation metrics

We rely on precision, recall, and F1-score to evaluate the accuracy of our anomaly detection method. These metrics are standard for detection problems with class imbalance (in our case, the number of anomalous cells is significantly smaller than that of normal cells).

Definition of true and false positives. The challenge in our case is to define what we consider as true positive (TP) and false positive (FP). Even when we inject anomalies in the controlled website scenario or synthetically in $\mathbb{M}$, we still have no complete ground truth. Visits to our controlled website may have suffered real issues that we have not introduced, and the dataset we use for introducing synthetic anomalies may likely contain real anomalies as well. These real anomalies appear as FP in our results. We consider a TP when our method correctly identifies a cell with an injected anomaly as anomalous. In the controlled website scenario, an injected anomaly may span two time bins. We consider a TP if the method detects at least one of the two time bins as anomalous. We are conservative and consider all other detected anomalies as FP.

Manual inspection of false positives. To evaluate the effect of real anomalies on the results we present, we manually inspect each website-metric time series and report cells that look anomalous, *i.e.*, a single large spike in the time series, or a group of consecutive bins with unusually high values. We then report additional values for precision and recall, which we label *M.I.* for Manual Inspection. In this case, if the method detects any of the manually reported anomalies, we do not count it as FP nor as TP. Note that we select only the cells that look the most anomalous. There are still cells that may be rightfully detected by the method but are not manually labeled as anomalous. Out of ~500k non empty cells, we only label 1,849 as anomalous (less than 0.4% of the entire dataset).

Controlled website. One issue with the controlled experiments is that we regularly introduce anomalies, and hence the anomalies may become part of the normal behavior. To avoid this bias, we conduct the evaluation in multiple steps each considering only few anomalies at a time. We remove all samples collected during injected anomalies except for the few anomalies we consider at a given step. At each step, we maintain the number of anomalous bins to 0.5% of the total number of bins (~11 anomalies per step). We repeat this process until we cover all injected anomalies and then compute the average recall and precision.

5.4 Baseline

We compare our method with the state of the art time series method for anomaly detection of web performance, Week-over-Week (WoW) [13]. As discussed in Section 2, WoW adopts a Seasonal-Trend Decomposition (STL) based approach, splitting time series into multiple components (trend, seasonal, and residual) to identify anomalies in search response times.

WoW works on individual time series, so we run it against one column of $\mathbb{M}$ at a time. Furthermore, WoW assumes that there are no empty time bins. To apply it to our dataset, we must fill in the gaps. We use linear interpolation on each per-column time series whenever there are no measurements during a one-hour bin. Given an empty bin at row i and column j of $\mathbb{M}$, we estimate $\mathbb{M}[i, j]$ as $\mathbb{M}[a, j] + (\mathbb{M}[b, j] - \mathbb{M}[a, j]) \times \frac{(i-a)}{(b-a)}$, where a and b indicate the closest non empty bins to i for column j, respectively before and after i.

WoW is designed for high volume of measurements and the authors filter out visits from mobile devices to reduce the variability of the measurements. In our already sparse dataset, mobile devices represents 52% of the visits. Thus, we keep these measurements (for the same reasons, we avoid splitting the dataset into different categories to limit the variability of the measurements).

6 VALIDATION RESULTS

To evaluate the accuracy of WMF, we use both the controlled experiments and the synthetic anomalies described in the previous section. We run WMF against the three-month dataset. We report results either in terms of precision-recall curves, or, where summarized values are needed, in terms of maximum F1 scores. Appendix C details how we tune the parameters of WMF.

Detection accuracy depends on anomaly duration and severity. We first evaluate the effect of anomaly duration and severity on the accuracy of our method. For this we rely on injecting synthetic anomalies. Intuitively, anomalies

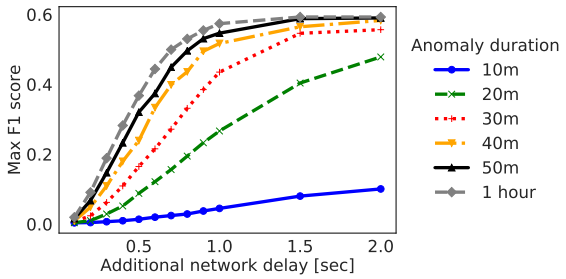


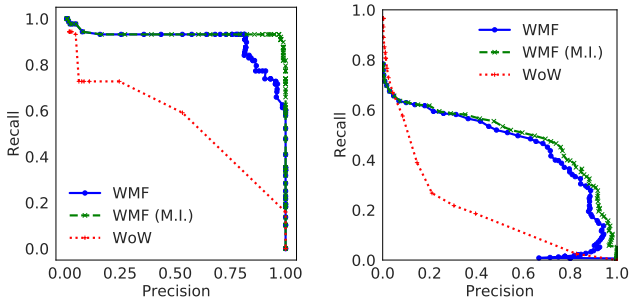
Figure 6— Effect of anomaly duration and severity on WMF's maximum F1 score.

that last for a long time and that significantly increase performance metrics should be easier to detect. Figure 6 presents the maximum F1 score of WMF on anomalies corresponding to added network delays between 0.1 and 2 seconds. Each line presents the results for different anomaly duration, and the maximum F1 score is calculated over different values of the detection threshold τ for each anomaly duration. Note that these numbers would be higher in practice, because our dataset contains true anomalies that our evaluation process is forced to treat as false alarms.

Figure 6 shows that WMF's performance increases with the severity of the anomaly, *i.e.*, additional network delay, as well as with the duration of the anomaly (in minutes). Anomalies with short duration (30-40 minutes or less) are hard to detect because our measurement time bins are 60 minutes in length. When the added network delay is less than about 500 milliseconds, detection is difficult overall; on the other hand, when additional network delay is 1 second and more, even relatively short anomalies show good values for the F1 score that can go up to 0.6 for long anomaly duration. We conclude that network delay corresponding to 1 second is an important performance region, and we focus on that next.

Precision and recall of WMF versus WoW. Focusing on the case of one-second network delay, we study the accuracy of WMF in more detail, and further compare it to that of WoW. In this case, we also use results from the controlled setting, and manually correct some of the false alarms in the analysis. Figure 7(a) presents the precision-recall curve of WMF and WoW for controlled anomalies, and Figure 7(b) presents corresponding results for synthetic anomalies. Curves are obtained by changing the different values of the anomaly detection threshold τ. The results show that WMF considerably outperforms WoW, for both controlled and synthetic anomalies. Considering the left plot, we see that WMF is highly effective on controlled anomalies, *i.e.*, anomalies where actual delays have been introduced into the network during the measured web transactions. Even without manually correcting for the anomalies already present in the dataset (blue curve), WMF is capable of achieving impressive precision and recall (above 80% for both metrics at the same time). When we manually inspect and remove anomalies that are already present in the data, WMF performs even better and can achieve above 90% in precision and recall at the same time.

In Figure 7(b), results span many different web sites which affects detection performance. Comparing the two plots, we see that injected delays appear to be more difficult to detect, but that WMF still strongly outperforms WoW. Further, we see that even across many diverse websites, WMF can achieve a good level of accuracy, with operating points that are above 50% in both precision and recall simultaneously.



(a)— Controlled website with 1 s extra delay.

(b)— Synthetic anomaly equivalent to 1 s extra delay.

Figure 7— Comparison of WMF and WoW.

Detection accuracy depends on the number of empty cells. One of the motivations behind the design of WMF is the need to handle websites with many empty measurement bins. Hence, we examine the effect of the number of empty bins on WMF's accuracy. We consider the synthetic anomalies scenario with 1 s additional network delay and we filter out columns based on the number of empty bins. Fig. 8 shows the maximum F1 score of WMF and WoW, when the maximum tolerated percentage of empty bins varies between 0% (only columns with no empty bins) and 100% (all the columns of matrix $\mathbb{M}$). The figure shows that, while F1 scores decrease as sites have more empty measurement bins, the performance of WoW is consistently lower than that of WMF. Further, the figure shows that even when considering columns with many empty bins (right side of the figure), WMF still achieves a high F1 score (around 0.6). Finally, the presence in the dataset of real anomalies not synthetically generated prevents achieving an F1 score closer to one.

7 WEB ANOMALIES IN THE WILD

In this section, we characterize the anomalies WMF detects on the three-month dataset described in Sec. 3.1 (without any controlled or synthetic anomalies). We first present the overall characteristics of these anomalies, then we conduct a more detailed analysis of a few anomalous events.

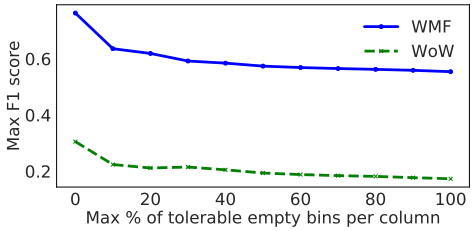


Figure 8— Effect of percentage of empty bins on the detection accuracy.

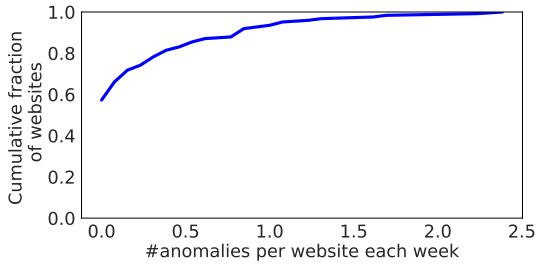


Figure 9— Frequency of the anomalies: average number of anomalies on each website per week.

7.1 Overview

WMF detects a total of 502 anomalies (cells in $\mathbb{M}$ with abnormal web performance) during the three-month measurement period, when using a threshold τ equal to the 99.9th percentile of $\mathbb{M}_R$.

Anomaly frequency. Figure 9 presents the average number of anomalies per week on each website. Approximately 60% of websites experience no anomaly during the three months of our study. In fact, the 502 anomalies we detect concentrate on only 56 out of the 125 websites. In particular, three websites experience at least one anomaly every five days. We also study the inter-arrival time of anomalies on a given website in Figure 10. We see that 20% of anomalies happen within ten hours or less of the previous anomaly. In those cases, it is likely that a single underlying issue lasts for several hours, but WMF may have only detected anomalies in a few of the bins (we will show one such example anomaly in the next section). Overall, this result suggests that the majority of anomalies in the dataset are isolated and short-term events – for instance, transient network congestion, a misconfiguration of the internal routing of a CDN, or a server overload caused by a spike in requests.

We compare the characteristics of the websites with and without anomalies, but found no particular factor that explains why a given website in the dataset experiences more anomalies. We analyze the number of empty bins in a website's data and the average number of visits, but neither has an effect.

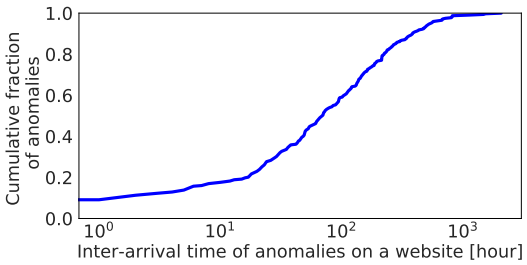


Figure 10— Inter-arrival time of anomalies on a website. While the majority of anomalies are far apart from each other, 20% of them are consecutive or within 10 hours of each other.

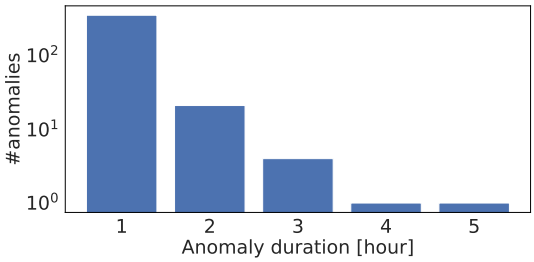


Figure 11— Anomaly duration (in hours).

Other factors may have an impact on the frequency of anomalies on a website, such as the complexity of the pages, as well as the optimization and the distribution of the resources across different servers. Unfortunately, we have no information about the back-end of the web services and part of the URL's are hidden, which prevents us from accessing all the webpages and develop further our analysis in this direction.

Anomaly duration. Figure 11 shows the distribution of the duration of anomalous events. For this analysis, we group consecutive anomalous bins together into individual anomalous events. Consistent with the results in Figure 10, the large majority of anomalies has a duration of one hour, which is the minimum size we can identify due to our bin sizes. A few anomalous events, however, last for four or five hours.

Website load and anomalies. One possible cause of anomalies can be website overload. We study the relationship between website load in terms of number of visits during the anomaly versus typical number of visits in Figure 12. We use min-max normalization to bring the number of visits in the anomalous bins between 0 (minimum number of visits reported in the column) and 1 (maximum number of visits). We see that in the majority of the cases, the anomalous bins have a smaller number of visits compared to the normal trend for the website. This result indicates that users may momentarily leave the website when performance degrades.

Anomalies per web performance metric. Some websites in

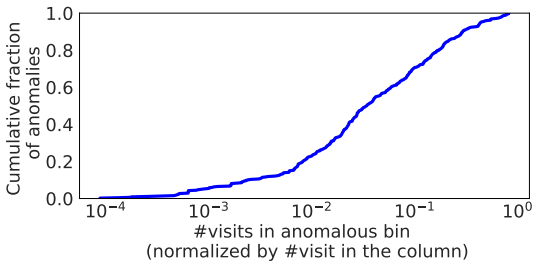


Figure 12— Number of visits in the anomalous bin (normalized by the number of visits in the column). In the majority of the cases, the number of visits decreases when there is an anomaly.

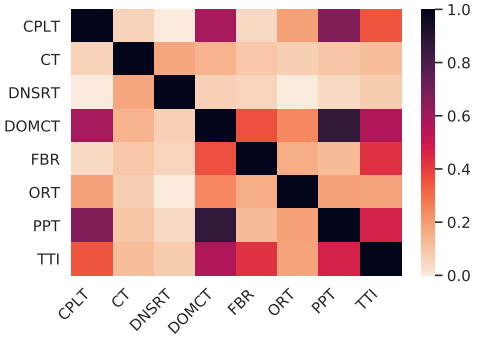


Figure 13— Pearson correlation between web metrics (columns of $\mathbb{M}$ of different metrics but same website).

our dataset report different metrics. The number of anomalies WMF detects on each metric is in line with the number of bins in $\mathbb{M}$ associated to each metric, with no type of measurement that seems to be more prone to anomalies.

Out of the 125 websites in our study, 63 collect more than one web performance metric. We can hence study the relationship between web performance metrics. Figure 13 shows the average Pearson's correlation between pairs of columns of $\mathbb{M}$ for the same website (we ignore rows where we have visits only on one of the columns). The DOM Completion Time shows moderate correlation with the Customized Page Load Time (0.59 average Pearson's coefficient), the Page Processing Time, and the Time To Interactive, which are all included in the DOMCT (see Figure 1). DOM Completion Time, however, presents very weak correlation with DNS Response Time and TCP Connection Time, which indicates that in general DOM Completion Time is dominated by the download of the page structure and objects.

Out of the 56 websites with anomalies, 41 collect multiple metrics. We also analyze the correlation between metrics when considering only the residual components. The same correlation exists between columns of $\mathbb{M}_R$. The residual component of the DOM CT still presents moderate correlation with PPT, TTI, FBR and Customized PLT. Yet, when WMF detects an anomaly on the DOM CT, it detects an anomaly on one of these other metrics for the same website in only 50% of the cases.

We can use the detection of an anomaly in one metric to improve the detection in other metrics and gain more confidence on the detection. For example, if we detect an anomaly on DOM CT in a time bin, we can look for anomalies in other metrics of the same website with a lower threshold τ. Considering, for instance, τ equal to the 95th percentile of $\mathbb{M}_R$ increases the percentage of agreements between metrics to 100%. We believe such cross-metric analysis to be a promising avenue for future work.

7.2 Analysis of anomalous events

Among the 502 anomalies WMF detects, we select four example events to study in more detail. We pick the most severe anomalies to allow analyzing the possible causes of the dysfunctions without the need for feedback from the operators of the websites. Some detections need indeed the operators knowledge to troubleshoot the anomaly.

Anomalous event#1, 13 hours of anomalous visits. On the 8th of October 2019, WMF detects two anomalies on a Chinese website within a five-hour time interval. We report in Figure 14 the Time To Interactive of visits to this website over the two days that include the detections. For a period of 13 hours, a fraction of the visits reports TTI values higher than 60 seconds, which increases by a factor of ten the number of visits that normally reach such a high value. This behavior does not correspond to any daily or weekly pattern (it is unique in the time series). Furthermore, we noticed that the majority of these anomalous TTI values fall within a relatively small time interval, between 62 and 70 seconds, which could be a symptom of timeouts expiring. We do not have any information about the back-end of the website, but we speculate that a problem affecting one of the servers hosting the website (or managing a specific resource) is behind this decrease in performance and possibly timeouts on some of the visits.

Anomalous event #2, national holiday. Figure 15 shows the time series of the Customized Page Load Time of a Spanish website, overlaid with the per-hour number of visits to the website. WMF detects three anomalies within 22 hours, on the 11th of October (highlighted with red dots in the figure). A sudden drop in the number of visits appears at the beginning of the detection period and lasts for 27 hours. This drop changes the normal composition of measurements within a cell of $\mathbb{M}$, causing the 75th percentile to abnormally vary from its normal behavior. While this shift corresponds to no daily

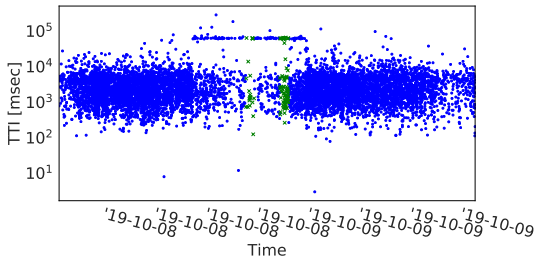


Figure 14— Anomalous event #1: TTI reported by each visit in a two days time interval, with green dots indicating the visits labeled as anomalous by WMF.

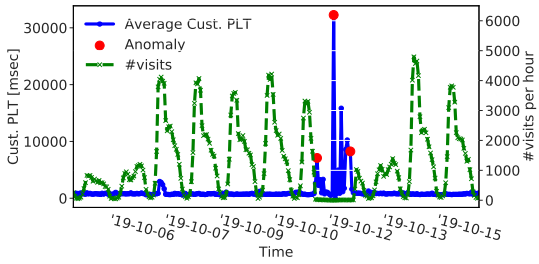


Figure 15— Anomalous event #2: sudden drop of visits during a national holiday.

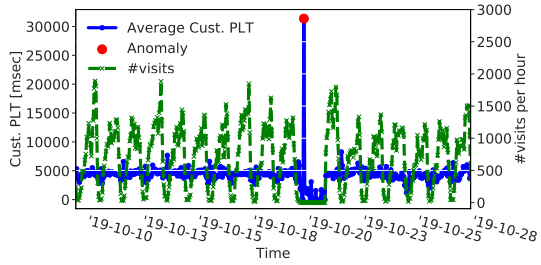


Figure 16— Anomalous event #3: website is likely do be down for one day.

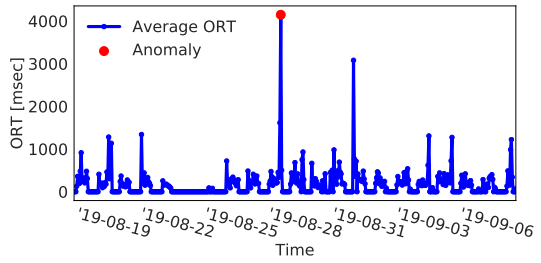


Figure 17— Anomalous event #4: large increase of Object Response Time for one single web page.

or weekly pattern, the 11th of October is a national holiday's eve in Spain, thus most likely the drop in visits is related to this festivity. We believe that the drop is caused by either a change in people's behavior during the holiday, or, by a maintenance scheduled on that day on the infrastructure of the webserver.

Anomalous event #3, WMF detects an anomaly on website likely to be down. Figure 16 shows the time series of the Customized Page Load Time reported by a commercial Italian website, overlaid with the number of visits. WMF reports an anomaly on the 20th of October on this website. Looking at the number of visits in that period, we see that the website experienced a significant problem and had the number of visits reducing to almost zero for 24 hours. Not having direct information from the administrators of the web service, we can only speculate that the website was most likely down for most of the day and the few users may have hit caches on the Internet or on their own devices, executing thus the Javascript code that performs the measurements even if the main website was not accessible. Note that measurement results are uploaded to the monitoring company, not the website.

Anomalous event #4, WMF detects a problem on a specific page. Figure 17 shows the time series of Object Response Time (ORT) on a Chinese website. In the three weeks period in the Figure, WMF detects few anomalies. Among those, it detects two consecutive anomalous bins with ORT higher than 5 seconds on the 28th of August. Among visits with high ORT, 98% are to a single page, which normally has an average ORT of 0.9 seconds. Considering all the visits within the two anomalous hours, the page covers 50% of the visits, while normally it covers about ~9%. We do not have access to the webpage, but we speculate that the page contains objects not present in other pages of the website and that users experienced problems in accessing these specific objects of the page, likely caused by the spike in the requests to the page itself. The company providing us with the dataset performs active measurements targeting the website and confirms a partial problem with the same website on the same day. Unfortunately, they store logs for limited period of time so we do not have further details.

8 CONCLUSION

We presented WMF, a web performance anomaly detection method that addresses the challenge raised by passive measurement across a wide range of general-purpose websites. Although individual websites may have few or no measurements at certain times, WMF leverages data from other websites to identify normal behavior (for example, daily and weekly patterns) and to detect abnormal web performance. WMF relies on normalization and aggregation of measurements of different webpages of a given website to increase the number of samples in individual time bins. WMF further leverages matrix factorization and a low rank representation

of the sparse dataset of web performance measurements to estimate the anomalous component of the performance of a website.

We validated WMF on a dataset of web performance measurements performed on 125 websites over a period of three months, with more than 290 million visits. In the controlled experiments on a single website, WMF achieved over 90% precision and recall. In the analysis of synthetic anomalies introduced in the dataset, the precision and recall were lower (as the dataset contains additional, unidentified anomalies); however WMF's F1 score was double that of state-of-the-art time series method. Finally, we characterized the anomalies in the dataset and showed that anomalies are often short, but that some anomalies can last for up to five hours and that some websites suffer from at least one anomaly every five days on average. Our case studies of four anomalous events illustrated that WMF can identify anomalous events with diverse characteristics. We conclude that WMF offers a valuable strategy that may be used by website administrators and third-party web monitoring companies to detect anomalies across the heterogeneous landscape of general-purpose websites.

Acknowledgements

This work was supported by the French National Research Agency under grant BottleNet no. ANR-15-CE25-0013 and by Inria within the Project Lab BetterNet. Mark Crovella was supported by NSF grant CNS-1618207, and by grants from Inria Paris, U. Sorbonne-Pierre et Marie Curie LIP6, and the Laboratory of Information, Networking and Communication Sciences (LINCS). We thank ip-label for providing us with the dataset and Benoit Boireau for his help in understanding ip-label's measurement methods and the collected dataset.

REFERENCES

[1] "State of online retail performance, 2017 holiday retrospective," https://www.akamai.com/us/en/multimedia/documents/report/akamai-state-of-online-retail-performance-2017-holiday.pdf, 2009, [Online; accessed 15-Nov-2019].

[2] J. Brutlag, "Speed matters for google web search," 2009.

[3] "How One Second Could Cost Amazon $1.6 Billion In Sales," https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales, 2012, [Online; accessed 17-Jan-2020].

[4] A. J., Brutlag Z and M. P., "Above the fold time: Measuring Web page performance visually Measuring webpage performance visually. ," http://conferences.oreilly.com/velocity/velocity-mar2011/public/schedule/detail/18692.

[5] "Speed Index," https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index, 2012, [Online; accessed 11-Oct-2019].

[6] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between qos metrics and web qoe using above-the-fold metrics," in *International Conference on Passive and Active Network Measurement*. Springer, 2018, pp. 31–43.

[7] Q. Gao, P. Dey, and P. Ahammad, "Perceived performance of top retail webpages in the wild: Insights from large-scale crowdsourcing of above-the-fold qoe," in *Proceedings of the Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2017, pp. 13–18.

[8] T. Hoβfeld, F. Metzger, and D. Rossi, "Speed index: Relating the industrial standard for user perceived web performance to web qoe," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2018, pp. 1–6.

[9] "Dynatrace," https://www.dynatrace.com/, [Online; accessed 17-Jan-2020].

[10] "Solarwinds Pingdom," https://www.pingdom.com/, [Online; accessed 17-Jan-2020].

[11] "IP-label," https://www.ip-label.co.uk/, [Online; accessed 17-Jan-2020].

[12] "Akamai," https://www.akamai.com/, [Online; accessed 17-Jan-2020].

[13] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 243–254.

[14] D. Liu, Y. Zhao, K. Sui, L. Zou, D. Pei, Q. Tao, X. Chen, and D. Tan, "Focus: Shedding light on the high search response time in the wild," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[15] S. A. Krashakov, A. B. Teslyuk, and L. N. Shchur, "On the universality of rank distributions of website popularity," *Computer Networks*, vol. 50, no. 11, pp. 1769–1780, 2006.

[16] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving user perceived page load times using gaze," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 545–559.

[17] "First Contentful Paint," https://developers.google.com/web/tools/lighthouse/audits/first-contentful-paint, [Online; accessed 30-Apr-2020].

[18] "Largest Contentful Paint, W3C Draft Community Group Report, 30 March 2020," https://wicg.github.io/largest-contentful-paint/, 2020, [Online; accessed 30-Apr-2020].

[19] R. B. Cleveland *et al.*, "Stl: A seasonal-trend decomposition procedure based on loess. 1990," *DOI: citeulike-article-id*, vol. 1435502.

[20] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[21] H. Z. Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *2008 International Symposium on Information Technology*, vol. 4. IEEE, 2008, pp. 1–6.

[22] A. H. Yaacob, I. K. Tan, S. F. Chien, and H. K. Tan, "Arima based network anomaly detection," in *2010 Second International Conference on Communication Software and Networks*. IEEE, 2010, pp. 205–209.

[23] R. Jašek, A. Szmit, and M. Szmit, "Usage of modern exponential-smoothing models in network traffic modelling," in *Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems*. Springer, 2013, pp. 435–444.

[24] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring." in *LISA*, vol. 14, no. 2000, 2000, pp. 139–146.

[25] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[26] M. Zhu, K. Ye, Y. Wang, and C.-Z. Xu, "A deep learning approach for network anomaly detection based on amf-lstm," in *IFIP International Conference on Network and Parallel Computing*. Springer, 2018, pp. 137–141.

[27] L. Huang, X. Nguyen, M. Garofalakis, M. I. Jordan, A. Joseph, and N. Taft, "In-network pca and anomaly detection," in *Advances in Neural Information Processing Systems*, 2007, pp. 617–624.

[28] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide

traffic anomalies," *ACM SIGCOMM computer communication review*, vol. 34, no. 4, pp. 219–230, 2004.

[29] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 117–132. [Online]. Available: https://doi.org/10.1145/1629575.1629587

[30] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. Gummadi, B. Krishnamurthy, and A. Mislove, "Towards detecting anomalous user behavior in online social networks," in *Proceedings of USENIX Security*, San Diego, CA, 2014.

[31] M.-R. Fida, E. Acar, and A. Elmokashfi, "Multiway reliability analysis of mobile broadband networks," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 358–364.

[32] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 20, 2008.

[33] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug 2009.

[34] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, p. 717, Apr 2009. [Online]. Available: https://doi.org/10.1007/s10208-009-9045-5

[35] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[36] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 2013, pp. 665–674.

[37] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 267–278. [Online]. Available: https://doi.org/10.1145/1592568.1592600

[38] L. Zhu, G. Wen, and S. Qiu, "Low-rank and sparse matrix decomposition with cluster weighting for hyperspectral anomaly detection," *Remote Sensing*, vol. 10, no. 5, p. 707, 2018.

[39] J. Clement, "Monthly search volume of bing as august 2017," https://www.statista.com/statistics/271640/bing-search-requests-by-country/, 2017, [Online; accessed 02-Dec-2019].

[40] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[41] P. Dixon, "Shopzilla Site Redesign – We get what we measure. Velocity 2009," http://assets.en.oreilly.com/1/event/29/Shopzilla%27s%20Site%20Redo%20-%20You%20Get%20What%20You%20Measure%20Presentation.ppt, 2009, [Online; accessed 15-Nov-2019].

A AGGREGATE FORM SELECTION

We report here the analysis of the impact that different data aggregation forms, *i.e.* average, median and other percentiles, have on the accuracy of our method. We use the controlled website scenario with anomalies created with one-second tc sessions and we build $\mathbb{M}$ summarizing the samples in a bin with different aggregation forms. Table 3 reports the F1 score that our method achieves in the different cases. We find the best aggregation to be the 75th percentile. The network problems we create in the controlled website scenario impact a variable percentage of visits within a bin (we randomly pick the start time of the tc session), thus aggregation forms such as the median or the 25th percentile of the average sometime fail to highlight the anomalous measurements. On the other end, the 99th percentile is prone to capture not only the anomalies, but also the noise caused by an handful of poorly connected devices.

Aggregate	Max F1 score
average	0.76
median	0.68
25th percentile	0.56
75th percentile	**0.88**
95th percentile	0.85
99th percentile	0.44

Table 3— Maximum F1 score on the controlled website when using different aggregation methods to build $\mathbb{M}$. One second additional network delay case.

B LOW RANK ESTIMATION METHOD SELECTION

In the literature there are different methods to reduce the rank of a matrix. Among the most common, we find ALS, LMaFit, SVD and PCA. We use the one second synthetic anomalies case and test each method in the task of computing $\mathbb{M}_L$.

While ALS and LMaFit tolearate empty bins, SVD and PCA do not. Thus, when we use the latter, we fill the gaps using interpolation (as described in Section 5.4). Table 4 reports the F1 score our method achieves in the three cases, with ALS having the highest score.

Low rank algorithm	Max F1 score
ALS	**0.52**
SVD	0.50
LMaFit	0.49
PCA	0.49

Table 4— Maximum F1 score for different low-rank matrix estimation methods (to compute $\mathbb{M}_L$). Synthetic anomaly equivalent to one second additional network delay.

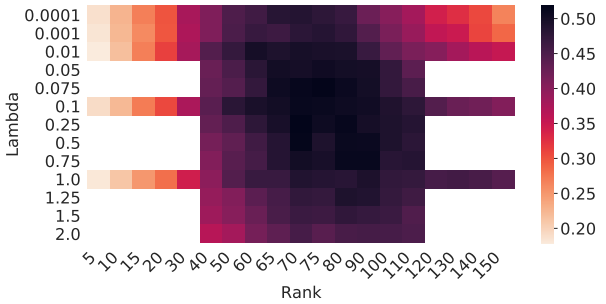


Figure 18— F1 score in detecting single-website anomalies with a web performance inflation given by $f(1s)$. We vary the ALS parameters lambda and rank to find the best setting on the training set of anomalies, corresponding to rank equal to 70 and lambda equal to 0.25.

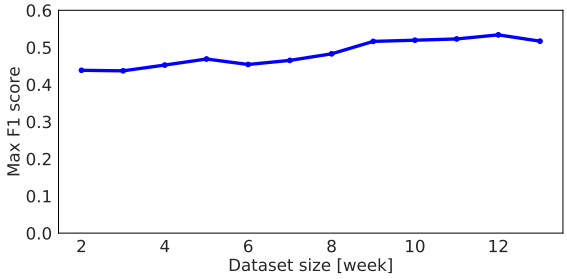


Figure 19— Max F1 score varying the dataset size from 2 weeks to the full 3 months.

C PARAMETER TUNING

In this section we describe the tuning process adopted to find the best setting for the ALS algorithm and for the dataset size. We consider a training set of anomalies created synthetically with 1-second and 2-seconds extra network delay — $f(1s)$ and $f(2s)$ — and evaluate the optimal settings to achieve the highest F1 score.

ALS parameters. To compute the low-rank approximation of $\mathbb{M}$ we first need to set the values for the regularization parameter of ALS, λ (see Equation 1) and for the desired rank for $\mathbb{M}_L$, k. Normally λ spans between 10^{-4} and 10^2, while k may go from 1 to the maximum possible rank of $\mathbb{M}$, 425. We vary the values for k and λ in Equation 1 and we report in Figure 18 the different F1 scores for the 1-second additional delay case (the 2-seconds case reports a similar pattern). Ranges of r and λ respectively of $[65-90]$ and $[0.1-0.75]$ achieves F1 scores higher than 0.5. In the rest of the evaluation we use r equal to 70 and λ equal to 0.25, which shows the highest F1 score.

Dataset size in time. The size in time of the datasets — the number of rows in $\mathbb{M}$— has an impact on the accuracy of the method *e.g.* measurements performed for less than a week fail to capture the weekend-week patterns. While in the evaluation in the core of the paper we always consider the total length of the dataset — 3 months — we report here the F1 score when varying the number of rows in the matrix. In order to avoid having to re-tune the method for each time window size we consider, we always adopt the best setting found for the 3 months case. We can see a stable trend of increasing F1 score when increasing the lenght of the dataset from 2 weeks to 2 months. Above the two months, the accuracy seems to be more stable. Adding more rows to the matrix does not seem to improve the ability of the method do find patterns among the different column and better define the normal behavior.

D SYNTHETIC ANOMALY FUNCTION

Here we present the study that allows us to calibrate the model function f needed to cast the delay increase into degradation of web performance, e.g., increase in the *TTI*. Figures 20(a), 20(b), 20(c) show the TTI of three distinct webpages during the different *tc*-sessions [1]. The red line corresponds to the prediction of the regression model we build with the measurements reported during the different *tc*-session.

For each webpage, w, we define the regression model as a function $f_w(d)$ that, given a desired extra delay, d, indicates the measurement value that a visit to the page would have if d was added to the communication. We use an Ordinary Least Square linear regression quadratic model and define the function in the form of $f_w(x) = \gamma x^2 + \alpha x + \beta$.

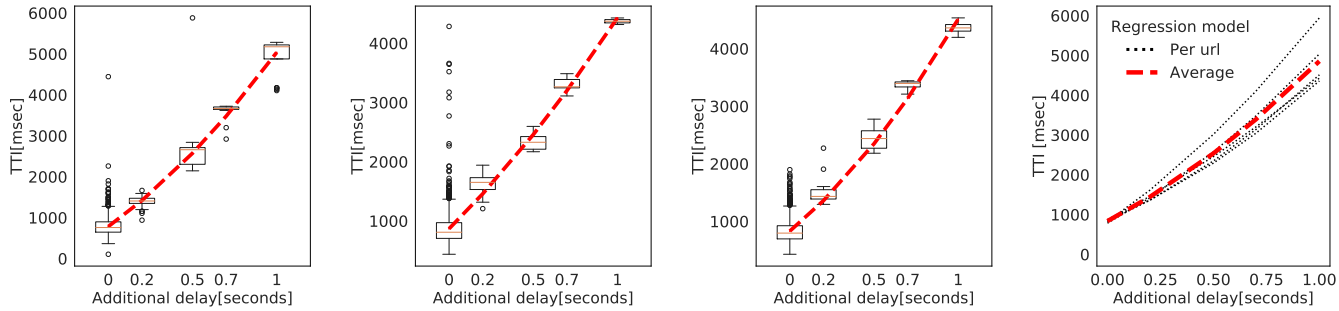
Figure 20(d) shows the different $f_w(d)$ for each of the five webpages, with an additional red line corresponding to the average of these degradations caused by the extra delay. We use the average degradation on this set of five webpages to approximate the effect of the extra delay on a generic page: we define the function, $f(t,d)$, that, given the performance of a visit, t, to a generic page, and the desired additional delay, d, indicates the performance the visit would have had if the additional delay d was applied:

$$f(t,d) = t \times \frac{\sum_{w \in W} \frac{f_w(d)}{f_w(0)}}{|W|},$$

where W is the set of webpages we build the regression model on, and $\frac{f_w(d)}{f_w(0)}$ corresponds to the degradation that d causes on w, with respect to the case with no extra delay (d equal to 0).

Now, given an additional delay of d seconds that we want to emulate and the recorded performance value of the visit whose performance measurement we want to inflate, we substitute t in the dataset with $f(t,d)$.

[1] We do not report some of the intermediate delay-levels in the plots in order to make the figures more readable and less crowded. All the levels are however used for the regression model.



(a)— Webpage with static content and one large size image.

(b)— Webpage with external resources retrieved by Javascript code.

(c)— Webpage with static content.

(d)— Regression models for the five pages. The red line corresponds to the average model.

Figure 20— Regression models to evaluate the impact of delay-based anomalies on the TTI on the controlled website. The increase of TTI is then used to artificially inflate the TTI of other websites for the synthetic anomalies study. The first three plots refer to different webpages, the fourth shows all the different models and the "by average" model.

Note that the goal of this analysis is simply to have reasonable multiplicative factors to inflate the web performance values as a function of the desired additional network delay, and thus have a set of known anomalies in the rest of the dataset. We do not argue that network delay affects distinct metrics or distinct websites in the same way. Also, we do not aim with this simple model to accurately capture the effects that network delay has on a generic website or web performance metric. We simply want to realize this inflation in a realistic way, while admitting that the TTI and the other web performance metrics strongly depends on the content of the page as well as on the network delay that affects them differently.