A Graph-based Framework for Real-time Vulnerability Assessment of Road Networks

Angelo Furno, N. E. El Faouzi
Univ Lyon, ENTPE, IFSTTAR, LICIT UMR_T9401
Lyon, France
Email: {angelo.furno, nour-eddin.elfaouzi}@ifsttar.fr

Rajesh Sharma
University of Tartu
Tartu, Estonia
Email: rajesh.sharma@ut.ee

Valerio Cammarota, Eugenio Zimeo
University of Sannio
Benevento, Italy
Email: zimeo@unisannio.it

***Abstract*—The ability to detect critical spots in transportation networks is fundamental to improve traffic operations and road-network resilience in smart cities. Real-time monitoring of these networks, especially in very large metropolitan areas, is a compelling challenge due to the complexity of computing robustness metrics. This paper presents a framework for identifying vulnerabilities in very-large road networks. The framework adopts graph-based modeling of road networks and exploits big-data techniques and technologies for processing such large and complex graphs. First, we use the framework to prove the existence of significant correlation between global efficiency and betweenness centrality. Then, we focus on an efficient algorithm, integrated in the framework, to rank the nodes according to this metric for finding potential vulnerabilities of a road network. To keep computation time under a "quasi" real-time threshold, a fast, requirement-driven, approximated strategy for computing betweenness centrality is adopted. The evaluation shows that the algorithm, integrated in the framework, exhibits a very good approximation for the most critical nodes, thus being well-suited for on-line operational monitoring.**

***Index Terms*—Smart Transportation, Network Resilience, Betweenness centrality, Contingency Analysis, Big Data.**

I. Introduction

Nowadays, cities are facing unprecedented challenges due to urbanization, climate change, and fast technology advances. They will be forced to cope with growing populations and connected (digital and physical) infrastructures whose robustness and resilience will become an essential property for their continuous operation.

Resilience describes the ability of a given system to provide fundamental services to people without discontinuity, even in presence of adverse or catastrophic events. As a consequence, it will be a key property for achieving smartness in cities of the near future. Resilience concerns several infrastructures used for transporting goods or people: energy, water, information and road networks are a few examples. An unforeseen event that originates a breakage in one of these infrastructures may cause incalculable damages with serious socio-economic consequences.

Transportation networks are bound to controllable (e.g., riots, technical errors, etc.) and uncontrollable events (e.g., earthquakes, floods, etc.) that can easily cause disruptions in the smooth operation of their services. In transportation literature, resilience is widely considered crucial to understand and quantify network robustness with respect to different kinds of failures and other unpredictable events [4], [21], [32], [33]. A robust transportation network is able reducing the impact of such failures on traffic flow, human safety and urban economy.

Vulnerability can be viewed as the flip-side of resilience: a vulnerable system has limited ability to absorb and react to the strains caused by adverse events. The availability of real-time information provided by a growing number of sensors and small devices distributed across geographic areas makes it possible today to use novel approaches to study robustness and adaptation ability of cities against possible unpredictable events, thus reducing their vulnerability. Moreover, the adoption of cloud computing and big-data techniques to effectively analyze this large amount of data paves the way to more quantitative studies to characterize city infrastructures. City key indicators derived from quantitative complex analyses will help planners and mobility operators to clearly identify vulnerabilities and adopt relevant strategies to mitigate risks. This is particularly important for physical transportation networks where accidents, weather conditions and social events create everyday risky situations for their correct operation.

In this paper, we present a graph-based framework that helps to identify the impact of breakage on very large road networks. The framework takes into consideration the existence of significant correlations between network properties and structural features of road networks, by focusing, in this work, only on the nominal state of the transportation network with correctly operating links and free-flow traffic conditions.

First, we analyze vulnerability using *Global Efficiency* (GE) [24], a network metric that has been widely used to characterize the overall resilience of power grids [22], the Internet [27], biological [1] and transportation [21] networks. Then, to spot the most vulnerable nodes inside the network we use *Betweenness Centrality* (BC), a fundamental metric of centrality already used in transportation to identify topological criticalities [4], [21], and traditionally preferred to other centrality metrics such as degree and closeness centrality [32]. BC has also been adopted for analyzing and predicting traffic flows in transportation networks [15], [19]. However, studying the impact of network breakage on road traffic remains a very complex task, difficult to be treated with traditional desktop hardware and state-of-the-art algorithms, especially in large metropolitan areas with a large amount of links and intersections.

The lowest time complexity for computing exact BC has been achieved by the algorithm proposed in [5]. Even though this algorithm is highly parallelizable, effective real-time vulnerability monitoring of large agglomerations requires much faster solutions on reasonable hardware. To that end, we propose the adoption in our framework of an approximated technique to compute BC. The algorithm has been proposed by the authors themselves in previous work [9], [10] to accelerate computation of BC, by keeping the error below a reasonable threshold depending on the specific requirements on computation time. In this paper, we integrate the algorithm in our proposed framework and prove the technique to be adequate for *real-time M-contingency analysis* of road networks due to its ability of tuning precision and performance.

M-Contingency analysis is a simulation-based technique that considers different qualitative assumptions over M likely events or phenomena in order to imagine different scenarios and come up with optimal responses under the analyzed circumstances. In the paper, the analysis is conducted on the road network of Lyon, France

The rest of the paper is organized as follows. In Sec. II, we present related work. In Sec. III, we describe our model and the metrics used to represent road-networks and their vulnerabilities, respectively. In Sec. IV, we present the framework for performing M-contingency analysis and in Sec. V the large-scale dataset used for testing the framework. Sec. VI shows the result of the evaluation of the framework and in particular of the algorithms adopted with the considered dataset. Sec. VII reports instead on a synthetic performance index that can be helpful for a user to configure the framework towards the desired behavior depending on the specific domain requirements. Sec. VIII concludes the paper by also discussing future directions.

II. Related Work

Various approaches have been proposed for quantifying the vulnerability of transportation networks, traditionally in off-line mode. Works include study of the impact of breakage in transportation networks in single-layer [21], multilayer [7] and multi-modal networks [30], by introducing random [7] or central (or critical) nodes failure [21]. Similarly, in a recent work [13], some authors of this paper have leveraged a mesoscopic fully fledged traffic simulator to perform contingency analysis for resilience evaluation, clearly showing the limitations in terms of computation time of traditional traffic simulation tools, which prove to be inappropriate for real-time vulnerability assessment, but much accurate in capturing complex traffic and user dynamics beyond basic topological network properties.

In graph-based modeling, one of the most used metric to calculate nodes' centrality is BC [8], especially in transportation network settings [32]. A large stream of transportation studies have exploited BC for traffic flow prediction (e.g., [2], [12], [14], [33]) or vulnerability quantification (e.g., [4], [21], [32]). In this context, some authors have highlighted limitations of the BC metric in representing traffic dynamics [12], [14], [16], [20]. However, such shortcomings can be overcome by augmenting the graph representation of the networks by taking into account also spatio-temporal aspects (e.g., congestion, accidents, road capacity changes, etc.) and geometric properties of road network [33], mapping them on a weighted dynamic graph. This additional information contributes to improve the effectiveness of the analysis but does not impact performance for searching relevant nodes in very large networks, which is the main objective of this work.

For the reasons above, we assume that road networks are in steady state and modeled as unweighted and undirected graphs. This assumption has been adopted also in other studies [4], [21], [32]. Some of them have tried to understand the effect of breakage, especially of nodes selected via BC, on transportation network in various parts of the world, such as Toronto [21], Melbourne [25], Sweden [17] and other metropolitan cities [4]. In comparison, no other study has been performed on Lyon metropolitan road network, which, *differently from most other studied cases, is a non-scale-free graph*[1] [26]. Moreover, *the datasets being used in previous studies are relatively small*, that is in the order of thousands of nodes. A larger size has an important impact on performance as the computation time for calculating BC is very high.

Very recently, some researchers have proposed different (exact or approximated) solutions to reduce the computation time of BC [3], [5], [29], [31]. In [23], an efficient algorithm for calculating BC has been proposed for incremental BC computation. However, the high speedup, characterizing the algorithm when one single node is inhibited, drastically reduces when analyzing the impact of M nodes. These incremental algorithms are very fast and useful when network changes are limited and continuous, but *M-contingency analysis is a much more complex and challenging problem as it should consider M different perturbations of the network to simulate possible critical scenarios as consequence of catastrophic events or phenomena*.

In conclusion, a proper contingency tool for vulnerability estimation in large-scale transportation networks is still lacking. In our study, we focus on reducing the computational effort required to compute network metrics for vulnerability analysis, in light of proposing a solution for on-line and continuous monitoring over large-scale networks.

III. Network modeling and metrics

In this section, we introduce the model adopted for representing road networks and the metrics used to measure their degree of vulnerability. Such representation and metrics are adopted as part of the framework for vulnerability assessment described in Sec. IV.

A. Network representation

We model transportation networks as undirected graphs $G(V, E)$, where V denotes nodes and $E \subseteq V \times V$ edges.

[1]A scale-free graph is characterized by a power-law degree distribution, i.e., a very limited set of nodes in the graph has a very large degree, while the majority of the nodes has only a few neighbors.

Each node v_i represents an intersection in the network and an edge e_{ij} between nodes v_i and v_j represents a road. $N = |V|$ denotes the number of nodes in the graph.

A path $p(v_i, v_j)$, between two nodes v_i and v_j, consists of a set of nodes and edges that connect these two nodes. If this set does not exist, the graph is disconnected in *separated components*. The length of a path between any two nodes v_i and v_j, represented by $len(p(v_i, v_j))$, is the number of edges (or hops) to reach v_j from v_i. If nodes v_i and v_j are directly connected, then the path length is 1. A shortest path between any two nodes v_i and v_j, denoted as $sp(v_i, v_j)$, is a path with the minimum number of hops among all the paths connecting the two nodes. Multiple shortest paths may exist between the same pair of nodes, i.e., all the paths having the same minimum number of hops. We denote as $\sigma_{v_i v_j}$ the number of shortest paths between v_i and v_j, while $\sigma_{v_i v_j}(v_k)$ represents the number of shortest paths from v_i to v_j that cross node v_k.

B. Nodes removal strategies

To analyze the impact of unpredictable events on network vulnerability, we can consider different nodes removal strategies, depending on the scenarios we intend to simulate for the analysis:

1) *Possible case*: the nodes are removed uniformly at random. This particular strategy captures real scenarios where an event can happen at any road or intersection causing traffic disruptions;
2) *Worst case*: removed nodes are the most critical according to a centrality index; they represent possible vulnerabilities of a road network.

The framework we propose in this paper is able to work in both the scenarios. However, for performing the correlation analysis between vulnerabilities and graph metrics, we target the worst case scenario.

As a metric to establish a criticality ranking of nodes, we consider BC [8], since it measures the number of shortest paths crossing a node. Nodes with higher BC correspond to central intersections from an infrastructural perspective, being usually selected by commuters to reach their destinations and naturally prone to breakdown compared to other nodes. The BC of a node v_k is defined as:

$$BC(v_k) = \sum_{v_i \neq v_k \neq v_j} \frac{\sigma_{v_i v_j}(v_k)}{\sigma_{v_i v_j}}, \tag{1}$$

where $\sigma_{v_i v_j}$ is the total number of shortest paths from node v_i to node v_j and $\sigma_{v_i v_j}(v_k)$ is the number of those paths that cross v_k. For scaling purpose, values are often normalized by dividing them by $N \cdot (N-1)$.

C. Resilience metrics

Network partitioning and network ability to move vehicles among nodes with a short number of hops might be useful metrics to evaluate the impact of network breakages on transportation networks. Therefore, to quantify the resilience of a road network, we consider the following metrics:

1) **Global Efficiency** (GE): represents the ability to efficiently exchange information in the network [24]. Let N be the total number of nodes in the network, and $len(sp(v_i, v_j))$ the length of the shortest path between any two nodes v_i and v_j, the GE of the network G is defined as:

$$GE(G) = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{len(sp(v_i, v_j))}. \tag{2}$$

2) **Vulnerability**: is defined as the drop in information exchange due to removal of a node and all corresponding edges [6]. Formally it is defined as:

$$V_{v_i} = \frac{GE - GE_{v_i}}{GE}, \tag{3}$$

where GE is the global efficiency of the original network as from Eq. 2 and GE_{v_i} is the global efficiency after the removal of node v_i and all the edges incident on it.

IV. On-line framework for contingency analysis

The framework that we propose in this paper aims at providing public administrators and maintenance enterprises with an easy-to-use tool for handling contingencies.

It has been developed as a multi-tier Web application and, according to this pattern, it is composed of four main logical tiers (see Fig. 1): (a) a Web-based client implemented atop AngularJS; (b) a front-end tier, exposing REST services to the client; (c) a business logic tier for supporting transactional accesses to data stored in the data-tier and for processing data when specific events occur, through Spark parallel jobs; (d) a data tier for handling graph-structured data, implemented atop Neo4J, and for storing unstructured or semi-structured data through HDFS.

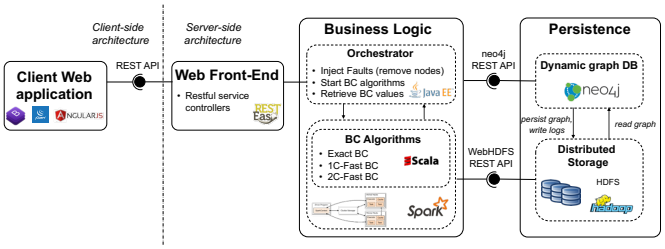


Fig. 1. Road Management Framework: multi-tier architecture

The framework Graphical User Interface (GUI) is presented in Fig. 2. The sidebar on the left enables a user (a) to filter nodes on the basis of their BC values, (b) to execute a job with the current configuration of potential faulty nodes and (c) to enable GUI auto-refresh. Additionally, it shows the number of filtered nodes and the status of the computation.

Faults, on the other hand, can be injected directly on the map, shown on the right side of the GUI. Here, each node is represented by a marker whose selection activates a popover to inject a fault. Several faults could be selected before triggering the computation of the new values of betweenness centrality through the processing of the underlying graph (job execution).

Marker colors map nodes to a discrete representation of BC, based on three levels: green, $0 \leq BC < x$; orange, $x \leq BC < y$; red, $BC \geq y$.

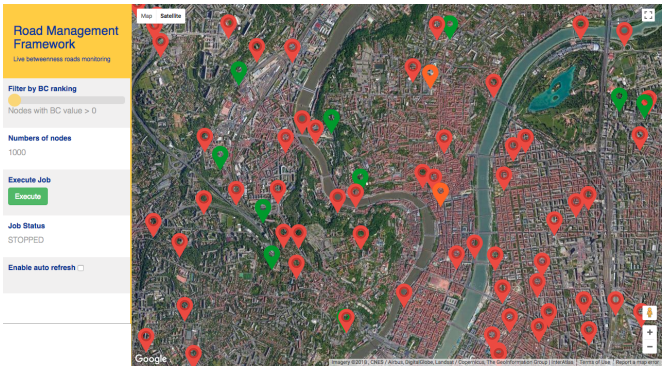


Fig. 2. Road Management Framework: GUI

By using the framework, a M-contingency analysis can be conducted, by identifying the potential critical nodes in case M hypothetical faults are injected into the network. Similarly, by exposing the REST API of the framework to external sensors, real faults could be captured and projected into the graph to analyze in quasi real-time the new network configuration. The dynamic interactions among the main software components of the system are described by the sequence diagram of Fig. 3.

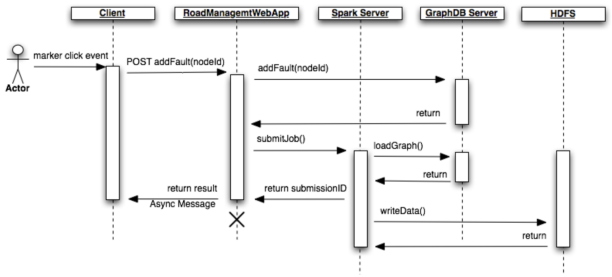
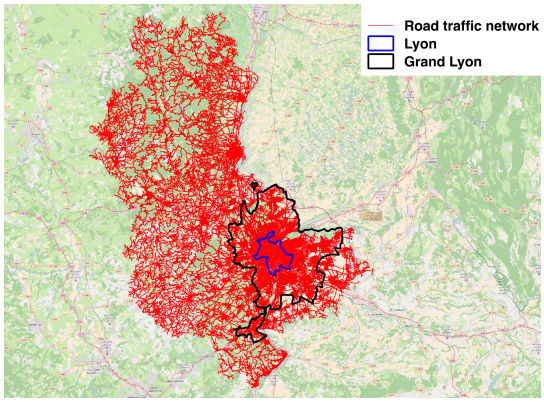


Fig. 3. Road Management Framework: dynamic diagram capturing the main interactions among the framework components.

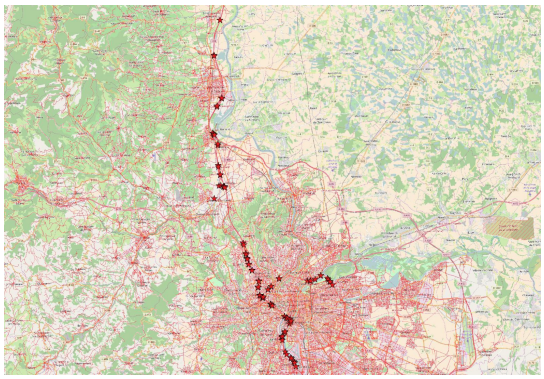
When a fault is injected into the system, the graph modeling the road network is changed and consequently a new processing job can be started to execute the algorithms proposed by the authors in [9], [10], whose code is written in Scala. The job is executed asynchronously on the Spark framework and, once terminated, the client refreshes the map to show the potential up-to-date vulnerabilities of the new network configuration.

V. Dataset

To test the framework with our algorithms for fast BC processing, we consider a graph corresponding to the road network of the Grand Lyon metropolitan area, France, and its surroundings, covering an area of approximately 3,000 Km^2 (see Figure 4a). This dataset was created using digital maps supplied by the French National Institute of Geographic Information (IGN). The network consists of 112,567 nodes and 240,372 edges.



(a) Map (in red) of the analyzed road network and its geographical extent

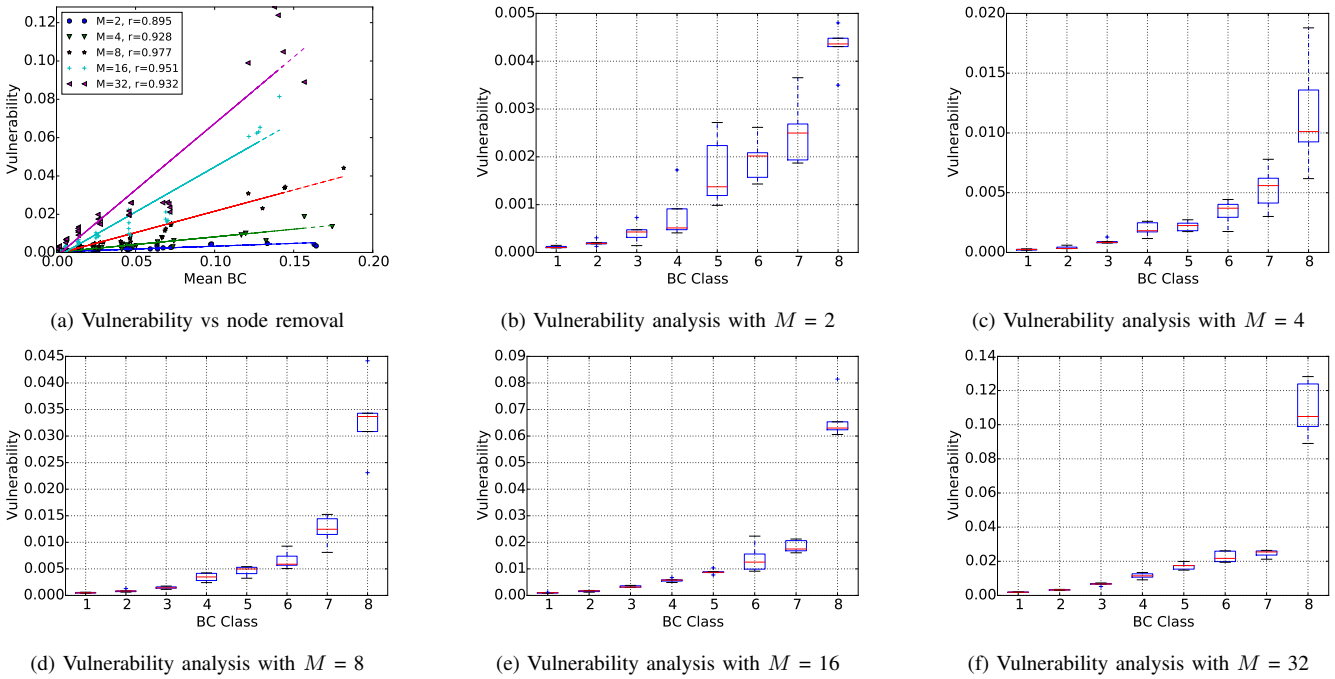


(b) Most critical nodes according to the BC metric (A6 and D383 highways)

Fig. 4. The Grand Lyon road traffic network.

In the original graph, some roads are split in multiple segments to capture properties that change along the road itself (e.g., street name, road slope, etc.). This means that a single long road with no real intersections along it can be actually composed of multiple nodes and edges in the graph. We decided to filter out this fine-grained information, by retaining only actual intersections and link endpoints as the nodes of our dataset. Next, to avoid a disconnected graph, we selected only the largest connected component (termed G_{Lyon} in the following) by filtering out very small isolated subnetworks that typically include country roads or cut areas lying at the border of the analyzed region. Finally, due to partial available information on road driving direction and number of lanes, we treat the network as undirected. We remark here that the solutions described in the next section can be easily extended to work with directed and weighted graph.

The final G_{Lyon} graph has 75,474 nodes and 96,406 edges. Nodes with the highest values of BC mostly correspond to highway interchanges (e.g., the A6 and D383) and roads that cross bridges, as reported in Fig. 4b. These represents elements of the network with limited alternative routes, and thus highly vulnerable spots from a topological perspective. The graph has a density of $3.38e^{-05}$ and clustering coefficient of 0.054. Its average path length and diameter are 84.74



(a) Vulnerability vs node removal

(b) Vulnerability analysis with M = 2

(c) Vulnerability analysis with M = 4

(d) Vulnerability analysis with M = 8

(e) Vulnerability analysis with M = 16

(f) Vulnerability analysis with M = 32

Fig. 5. Impact of M nodes removal from different BC-ranges on network vulnerability

and 244, respectively. The graph is therefore very sparse and characterized by a Gaussian distribution of the node degree, with an average value of 2.55.

VI. Evaluation

We evaluated the framework with our Scala algorithms [9], [10] for fast computation of BC, by leveraging multi-core processing for parallel execution. Apache Spark was configured to work in local mode, using 10 threads to partition the execution load of the map-reduce tasks on the available cores of an Intel Xeon E5 2.4 GHz multi-core machine, equipped with 56 virtual cores and 128 GB of DDR4 RAM. We also considered a Scala implementation of Brandes' algorithm [5] (referred as *Exact-BC*), used as a benchmark in performance evaluation. *Exact-BC* was executed in the same testing environment used for *Fast-BC*, with 10 threads for parallelism.

A. Betweenness centrality for vulnerability assessment

In this section, we use the framework to verify and quantify the presence of correlation between *vulnerability* (as from Eq. 3) and nodes' betweenness centrality on the G_{Lyon} network, the latter values computed with the Exact-BC algorithm.

The framework is leveraged to simulate network failures, by removing M nodes from the graph and by measuring the new values of the vulnerability metric after nodes removal. As a criterion to select nodes for removal, we consider the descending order of BC (i.e., worst-case contingency). Since G_{Lyon} is characterized by many nodes with very close values of BC, to better verify the correlation between vulnerability and BC, we group the nodes of the graph in multiple ranges of BC, via Jenks natural breaks classification method [18].

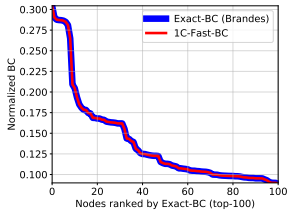
The number of ranges considered in our analysis is equal to eight[2]. Hence, we perform five tests by removing M (M= 2, 4, 8, 16 and 32) nodes, selected uniformly at random from each of the ranges. Nodes are inserted back in the graph after each test with a specific BC range. We repeat each test several times to make our results statistically relevant.

Fig. 5a shows the scatter plots of the impact of M-nodes removal from the eight classes (mean BC of the M removed nodes on the X-axis) on vulnerability (Y-axis), where M is represented by different colors and markers, and a linear regression shows the relative trend. As expected, vulnerability increases sharply as mores nodes (i.e., larger M) are removed from each class. This is especially evident in those classes with higher BC values. The linear regression for each of the class shows the linear behavior with different values of slope.

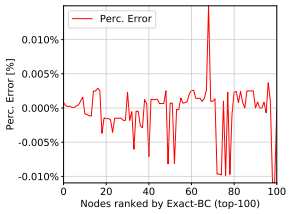
Figures 5b:5f plot vulnerability for $M = 2$, 4, 8, 16 and 32, respectively. In each one of these micro-analyses, we observe a similar pattern by using box plots. The analysis clearly shows that removing nodes with BC in higher classes (on the right side of the X-axis) increases vulnerability more than removing nodes in lower BC classes.

Nodes with higher values of betweenness centrality are the ones that affect most vulnerability of a road-traffic network. Monitoring should continuously identify these nodes since, if disrupted, they may significantly reduce network connectivity. On the other hand, their inhibition may significantly alter network topology, forcing to recompute betweenness centrality for the whole network.

[2]Other numbers of ranges have been tested with similar results.

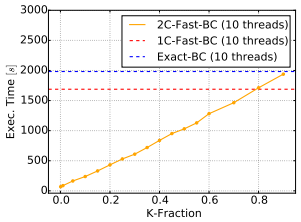


(a) Values of normalized BC with 1C-Fast-BC and Exact-BC for the top-100 nodes

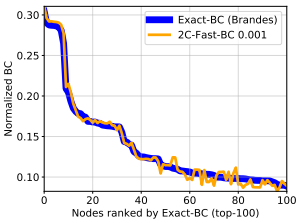


(b) Percentage error of 1C-Fast-BC over the top-100 nodes

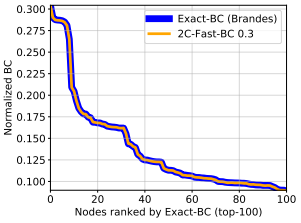
Fig. 6. Accuracy of the 1C-Fast-BC algorithm



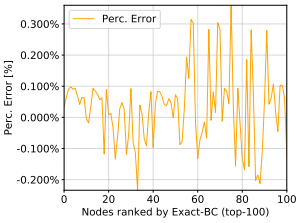
(a) Execution time of 2C-Fast-BC with different K-Fractions



(b) BC values with K-Fraction = 0.001 for the top-100 nodes



(c) BC values with K-Fraction = 0.3 for the top-100 nodes



(d) Percentage error with K-Fraction = 0.3 sover the top-100 nodes

Fig. 7. Performance Evaluation of the 2C-Fast-BC algorithm

B. *M-contingency analysis: approximated fast BC*

We tested the framework in two different operation modes, based on two different algorithms aimed at rapidly computing approximate BC values. The first mode, called *1C-Fast-BC*, exploits topology-based clustering to reduce the number of breadth-first-searches performed by the Brandes' algorithm. This strategy has been originally proposed in [29] by the authors. The second mode, called *2C-Fast-BC*, exploits an algorithm aimed at further reducing execution time on non-scale-free graphs, by preserving acceptable levels of accuracy in BC approximation. This behavior is desirable when computation time must be very low and errors are tolerated below a specific threshold. The *2C-Fast-BC* algorithm extends the one used with *1C-Fast-BC* by means of an additional clustering procedure, implemented via a Map-reduce parallel version of K-means clustering, based on the implementation provided in Apache Spark *MLlib* machine learning library. The *2C-Fast-BC* algorithm has been originally presented in [9], [10] by the authors, and it has proved to allow for a quicker computation of BC values than *1C-Fast-BC*, at the price of larger error.

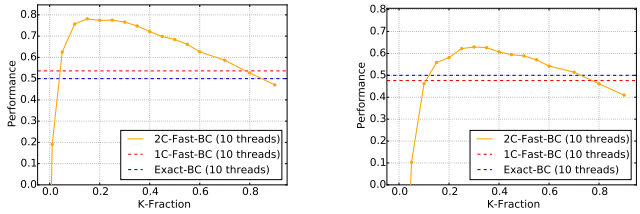
It is worth to remark that, as a consequence of the adopted algorithms, the *1C-Fast-BC* mode corresponds to a parameter-free configuration of the framework, while the *2C-Fast-BC* one requires the setting of one single parameter (i.e., *K-fraction*) in the range [0, 1], where 0 means tolerating a larger approximation error in order to achieve the smallest possible computation time. Conversely, a K-fraction of 1 drives the algorithm towards the most precise approximation (the same as provided by *1C-Fast-BC*), but with a larger execution time.

In the following, we briefly report on the performance trade-off characterizing the *2C-Fast-BC* algorithm, compared to the *1C-Fast-BC* and the *Exact-BC*, evaluated in the G_{Lyon} case study. The interested reader may refer to our previous work [9] for a more detailed description of the algorithms, as well as their evaluation on different kinds of graphs (e.g., scale-free, small world, random).

Given the specific conceptual design of the proposed algorithms and our objective of continuously monitoring only the most critical intersections of the road-traffic network, the reported performance analysis focuses on the most-critical nodes of the network (e.g., top-100, top-5%, etc.). However, it is worth to remark that BC is computed for all the nodes of the network.

Fig. 6 shows the normalized BC values computed by the *1C-Fast-BC* and *Exact-BC* algorithms for the top-100 nodes, ranked according to their exact value of BC. Fig. 6a highlights an almost perfect overlap of the two curves, confirmed by the extremely low percentage error in Fig. 6b, bounded in the range [-0.015%, 0.015%]. In other words, the *1C-Fast-BC* algorithm produces the exact same results (i.e., identification of the most critical nodes) reported in Fig. 4b for *Exact-BC* on the G_{Lyon} graph. However, the algorithm takes 1,688 seconds to complete, a fairly limited improvement with respect to the 1,982 seconds of the *Exact-BC*.

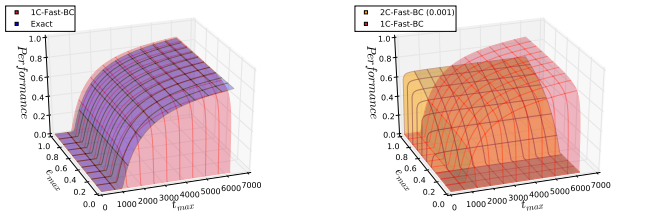
Fig. 7a shows that *2C-Fast-BC* execution time stays below both *1C-Fast-BC* and *Exact-BC* (that do not depend on the K-fraction parameter), up to very large values of K-Fraction (i.e., 0.8). Larger values of the parameter result in higher execution times (larger than the ones of *1C-Fast-BC* for K-Fraction > 0.8) due to the computation overhead associated to K-means clustering. Obviously, lower K-Fractions introduce a larger approximation that negatively affects BC values even for critical nodes. As an example, Fig. 7b presents BC values for the top-100 nodes when using an extremely low K-Fraction of 0.001. For this configuration, computation time is extremely low (69 s) but percentage error is relatively high (i.e., in the range [-12%, 12%]). Conversely, by choosing slightly larger K-Fractions (e.g., a K-Fraction of 0.3), the algorithm quickly converges towards very good levels of accuracy, as highlighted by the perfect overlap with the *Exact-BC* values in Fig. 7c. Such improved approximation is confirmed by a percentage error in the range [-0.2%, 0.3%] (see Fig. 7d). As in the previous case of *1C-Fast-BC*, the most critical nodes reported



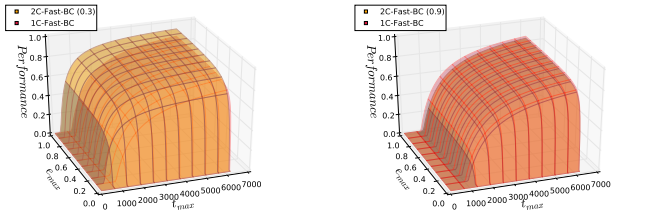
(a) e_{Max} = 0.25, t_{Max} = 1982s (b) e_{Max} = 0.1, t_{Max} = 1982s

(c) e_{Max} = 0.05, t_{Max} = 1982s (d) e_{Max} = 0.25, t_{Max} = 600s

Fig. 8. Performance



(a) Exact-BC vs 1C-Fast-BC (b) 2C-Fast-BC (K-Fraction 0.001) vs 1C-Fast-BC



(c) 2C-Fast-BC (K-Fraction 0.3) vs 1C-Fast-BC (d) 2C-Fast-BC (K-Fraction 0.9) vs 1C-Fast-BC

Fig. 9. 3-d Performance curves with different values of t_{max} and e_{max} (t_{ideal} = 0s)

in Fig. 4b were correctly identified by the *2C-Fast-BC* on the G_{Lyon} graph. Most importantly, the *2C-Fast-BC* with a K-Fraction of 0.3 executed in only 609 seconds, corresponding to a 277% and a 325% decrease in execution time with respect to the *1C-Fast-BC* and *Exact-BC*, respectively.

VII. An Aggregate Index of Performance for Framework Configuration

Since the *2C-Fast-BC* algorithm generates approximated BC values in a time depending on the error we can tolerate, an aggregated index for evaluating both performance and accuracy is useful. To this end, we propose the following:

$$P(x) = 0.5\left(1 - \frac{t_x - t_{ideal}}{t_{Max} - t_{ideal}}\right) + 0.5\left(1 - \frac{e_x}{e_{Max}}\right) \quad (4)$$

The equation depends on two variables that are computed after executing *2C-Fast-BC* in configuration x: 1) the time taken (t_x) for BC calculation; 2) the NRMSE error[3] (e_x) of the approximated BC values. The equation includes three parameters t_{ideal}, t_{Max} and e_{Max} that can be fixed according to the specific domain requirements. Specifically, t_{Max} is the maximum tolerated execution time. Similarly, e_x represents the maximum allowed NRMSE. Finally, t_{ideal} constitutes the minimum amount of time allowed to the execution of the algorithm. The idea is to compute $P(x)$ during a profiling phase on a given network, where the framework is tested in different modes and with several values of K-fraction.

In the first setting of the P metric, represented in Fig. 8a, we defined t_{ideal} equal to the execution time of the fastest *2C-Fast-BC* configuration, i.e., 60 seconds with a K-Fraction of 0.001. We set t_{Max} to the execution time of the Exact-BC algorithm (i.e., 1,982 seconds) and e_{Max} to 0.25. This setting rewards computing times below the one of Exact-BC and errors below 25% NRMSE. Fig. 8a clearly shows the values of K-Fraction satisfying these performance requirements, i.e., K-Fraction should be selected within the range [0.05, 0.7], with the most performing configuration being associated to a K-Fraction of 0.15. A comparison with the performance of *Exact-BC* and *1C-Fast-BC* is visually reported in the figure.

Different settings of e_{Max} and t_{Max} are depicted in Figs. 8b:8d. Such settings correspond to more stringent requirements on the tolerated NRMSE (Figs. 8b and 8c) and execution time (Fig. 8d), respectively. Interestingly, a configuration with an extremely low tolerated execution time of 600 seconds and maximum NRMSE of 0.25 (Fig. 8d) can be satisfied only by using the 2C-Fast-BC with a K-Fraction between 0.001 and 0.25.

In Fig. 9, we exploit 3D plots to represent the surfaces generated by varying the two parameters t_{Max} and e_{Max} in Eq. 4, for both 1C-Fast-BC and 2C-Fast-BC. Fig. 9a compares the performance of 1C-Fast-BC to that of Exact-BC. The two surfaces are mostly overlapping, thus showing very similar performance on the G_{Lyon} graph. In particular, performance is slightly better with 1C-Fast-BC when a larger maximum error (i.e., larger than 0.4) can be tolerated. Exact-BC should instead be preferred whenever the maximum tolerated time is large enough (i.e., larger than 1600 seconds) for the algorithm to complete, and the maximum tolerated error is smaller than 0.2. Different configurations of 2C-Fast-BC are compared to 1C-Fast-BC in Figs. 9b:9d. Specifically, Fig. 9b is related to the lowest values of K-Fractions (i.e., 0.001) considered in our evaluation. Good performance can be achieved with this

[3]The NRMSE is defined as: $\frac{1}{\bar{\sigma}}\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$, with $\bar{\sigma}$ denoting the mean of the exact BC values and e_i representing the difference between exact and approximated values of BC for node i. NRMSE is generally preferred to the percentage error when 0-values are present among the expected ones.

configuration when the maximum tolerated execution time is significantly low (e.g., lower than 100 seconds), and a fairly large error on BC values can be tolerated (e.g., larger than 0.5-0.6). Configurations with larger K-Fractions (e.g., 0.3 in Fig. 9c) appear to be generally preferable when requirements are not particularly stringent both on BC error and computation time. Finally, configurations with larger K-Fractions tend to exhibit performance that are very close to that of 1C-Fast-BC (see Fig. 9d).

VIII. Conclusion

We have presented a big-data framework and its tuning capabilities for performing real-time M-contingency analysis. The framework is based on fast algorithms for computing betweenness centrality in a graph-based representation of road networks. This metric, in fact, has proved to be strongly related to vulnerabilities, identified via the decrease of the global efficiency of a network.

The results are very promising since the algorithms, on the basis of a tolerated approximation, are able of rapidly locating the most M critical nodes of a faulty road network.

In order to generalize the approach to dynamic networks, three main extensions are planned: 1) adapting the algorithms for fast BC computation to weighted and directed graphs; 2) enriching network modeling by taking into account additional properties of road networks (e.g., road length and capacity, number of lanes, land use information [11]), as well as dynamic traffic data (e.g., demand, accidents, travel times, etc.); 3) studying the impact of nodes removal in multi-modal transportation networks modelled as multilayer networks [28].

Moreover, the framework will be extended with features to: 1) collect heterogeneous data from sensors; 2) adaptively change the settings of the 2C-Fast-BC algorithm according to dynamically evolving domain requirements (e.g., the framework could run in an *emergency mode*, requiring low computation time and larger tolerated error, or alternatively in a *maintenance mode*, demanding smaller error but allowing for more execution time).

References

[1] S. Achard and E. Bullmore. Efficiency and cost of economical brain functional networks. *PLoS Computational Biology*, 03(02), 2007.

[2] Y. Altshuler, R. Puzis, Y. Elovici, S. Bekhor, and A. BaglioniPentland. Augmented betweenness centrality for mobility prediction in transportation networks. *Finding Patterns of Human Behaviors in Network and Mobility Data (NEMO)*, 2011.

[3] M. Baglioni, F. Geraci, M. Pellegrini, and E. Lastres. Fast exact computation of betweenness centrality in social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 450–456, 2012.

[4] B. Berche, C. von Ferber, T. Holovatch, and Y. Holovatch. Resilience of public transport networks against attacks. *THE EUROPEAN PHYSICAL JOURNAL B*, 71(1):125–137, 2009.

[5] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(163), 2001.

[6] L. Costa, F. Rodrigues, G. Travieso, and P. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.

[7] M. De Domenico, A. Solé-Ribalta, S. Gómez, and A. Arenas. Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, 2014.

[8] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 25:35–41, 1997.

[9] A. Furno, N. E. El Faouzi, R. Sharma, and E. Zimeo. Reducing pivots of approximated betweenness computation by hierarchically clustering complex networks. *International Conference on Complex Networks and Their Applications*, 2017.

[10] A. Furno, N. E. El Faouzi, R. Sharma, and E. Zimeo. Two-level clustering fast betweenness centrality computation for requirement-driven approximation. *IEEE Big Data 2017 Conference*, 2017.

[11] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda. A tale of ten cities: Characterizing signatures of mobile traffic in urban areas. *IEEE Transactions on Mobile Computing*, 16(10):2682–2696, 2017.

[12] S. Gao, Y. Wang, Y. Gao, and Y. Liu. Understanding urban traffic-flow characteristics: a rethinking of betweenness centrality. *Environment and Planning B: Planning and Design*, 40(1):135–153, 2013.

[13] P. Gauthier, A. Furno, and N. E. El Faouzi. Road network resilience: how to identify critical links subject to day-to-day disruptions? *97th TRB Annual Meeting*, 2018.

[14] P. Holme. Congestion and centrality in traffic flow on complex networks. *Advances in Complex Systems (ACS)*, 06(02):163–176, 2003.

[15] A. Jayasinghe, K. Sano, and H. Nishiuchi. Explaining traffic flow patterns using centrality measures. *International Journal for Traffic & Transport Engineering*, 05(02):134–149, 2015.

[16] A. Jayasinghe, K. Sano, and H. Nishiuchi. Explaining traffic flow patterns using centrality measures. *International Journal for Traffic & Transport Engineering*, 5(2):134–149, 2015.

[17] E. Jenelius and L.-G. Mattsson. Road network vulnerability analysis of area-covering disruptions: A grid-based approach with case study. *Transportation Research Part A: Policy and Practice*, 46(5):746–760, 2012.

[18] G. F. Jenks. The data model concept in statistical mapping. *International Yearbook of Cartography*, 7:186–190, 1967.

[19] A. Kazerani and S. Winter. Can betweenness centrality explain traffic flow? *International Conference on Geographic Information Science*, 2009.

[20] A. Kazerani and S. Winter. Can betweenness centrality explain traffic flow? *AGILE*, 2009.

[21] D. King and A. Shalaby. Performance metrics and analysis of transit network resilience in Toronto. *Transportation Research Record*, 2016.

[22] R. Kinney, P. C. , R. Albert, and V. Latora. Modeling cascading failures in the north american power grid. *The European Physical Journal B - Condensed Matter and Complex Systems*, 46(1):101–107, 2005.

[23] N. Kourtellis, G. D. F. Morales, and F. Bonchi. Scalable online betweenness centrality in evolving graphs. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 00:1580–1581, 2016.

[24] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Phys. Rev. Lett.*, 87(19), Oct. 2001.

[25] G. Leu, H. Abbass, and N. Curtis. Resilience of ground transportation networks: a case study on melbourne. *33rd Australasian Transport Research Forum Conference*, 2010.

[26] K. Ohara, K. Saito, M. Kimura, and H. Motoda. *Accelerating Computation of Distance Based Centrality Measures for Spatial Networks*.

[27] C. R. Palmer, G. Siganos, M. Faloutsos, C. Faloutsos, and P. B. Gibbons. The connectivity and fault tolerance of the internet topology. *ACM SIGMOD/PODS Workshop*, 2001.

[28] R. Sharma, M. Magnani, and D. Montesi. Investigating the types and effects of missing data in multilayer networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 392–399, 2015.

[29] P. Suppa and E. Zimeo. A clustered approach for fast computation of betweenness centrality in social networks. In *2015 IEEE International Congress on Big Data*, pages 47–54, June 2015.

[30] M. A. P. Taylor, G. D'este, et al. Concepts of network vulnerability and applications to the identification of critical elements of transport infrastructure. 2003.

[31] J. Yang and Y. Chen. Fast computing betweenness centrality with virtual nodes on large sparse networks. *PLoS One*, 2011.

[32] Y. Zhang, X. Wang, P. Zeng, and X. Chen. Centrality characteristics of road network patterns of traffic analysis zones. *Transportation Research Record: Journal of the Transportation Research Board*, 2256:16–24, 2011.

[33] S. Zhao, P. Zhao, and Y. Cui. A network centrality measure framework for analyzing urban traffic flow: A case study of wuhan, china. *Physica A: Statistical Mechanics and its Applications*, 478:143 – 157, 2017.