Fast Computation of Betweenness Centrality to enable Real-time Resilience Assessment and Improvement of Complex Transport Networks

Cecile Daniel[1], Angelo Furno[1], Nour-Eddin El Faouzi[1], Rajesh Sharma[2], and Eugenio Zimeo[3]

[1]Univ. Lyon, ENTPE, Univ. Gustave Eiffel, LICIT, F-69518, Lyon, France
[2]University of Tartu, Institute of Computer Science, Juhan Liivi 2, Tartu, Estonia
[3]University of Sannio, Department of Engineering, 82100, Benevento, Italy

Extended abstract submitted for presentation at hEART 2020 9th Symposium
Word count: 2997 words

1 Introduction

With the growth of the population concentrated in urban areas of large agglomerations, the need for efficient and resilient multi-modal transportation systems is paramount. To model, analyze and improve transportation dynamics at large scale, complex networks represent an extremely versatile toolkit: multi-modal mobility networks can be modelled as a multi-layered weighted graph.

In the last decade, several works [1, 2, 3] have shown that complex network approaches based on computation of centrality metrics can be extremely useful to model and analyze the resilience properties of complex networks.

In such representation, each layer of the graph can be associated to a transportation mode (e.g., road, metro, buses, etc); each node of the network is an intersection between roads, a parking spot or a bus/metro stop/station; and the edges are links between the nodes, possibly belonging to different layers of the transportation network (e.g., links connecting bus with metro stations or parking areas in the proximity of a road intersection with bus stops, etc.).

In this graph-based model, it is also possible to associate weights to the network edges, e.g., (average) travel time or, alternatively, speed/flow measurements retrieved via sensors (e.g., floating car data, loop detectors, cameras, etc.) and matched to the links of the network. We could also consider the possibility that weights dynamically evolve in time, as it could be the case in a future-generation smart city, equipped with an online monitoring system capable of continually collecting traffic speed observations at city- or region-scales (e.g., via navigation systems, mobile phone apps, etc.).

Based on such model, in the rest of the paper, the mobility network and its dynamics are therefore described by a temporal, (multi-layer) graph, composed of multiple snapshots (e.g., graph instances), each associated to a different weight configuration related to a specific period of observation (e.g., every 15 minutes). For simplicity reasons, however, in this preliminary work, we only consider one transport mode, i.e., the road network, while the reported conclusions can generalize rather easily to a multi-modal graph, as we intend to demonstrate in our future work.

Specifically, in this paper we focus on betweenness centrality (BC), a first-choice indicator to analyze the resilience properties of complex networks. BC has been used, e.g., to identify topological bottlenecks in metro systems [4], road networks [5] and even multi-modal systems [6].

For a graph G with a set of nodes V and set of edges E such as $n = |V|$ and $m = |E|$, the betweenness centrality of a node v in the graph is defined as follows:

$$BC(v) = \sum_{s,t \in G} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

Where σ_{st} is the number of shortest paths from source s to destination t and $\sigma_{st}(v)$ is the number of shortest paths from s to t going through node v. To efficiently compute BC, the previous formula is split in contributions from different *source nodes*, defined as $\delta_{s\bullet}(v) = \sum_{t \in G} \frac{\sigma_{st}(v)}{\sigma_{st}}$, such as:

$$BC(v) = \sum_{s \in G} \delta_{s\bullet}(v) \tag{2}$$

Some authors have proposed variations of the BC metric taking into account travel demand information [7], time-varying traffic dynamics [8] or geometric properties of the the road network [9]. In all these work and many other from the literature, BC is recognized as a very important metric from a transportation perspective, whose computation can unveil fundamental properties of the represented system.

The computation of BC is however cumbersome. Based on Eq. 2, for each node of the network it is necessary to perform a single-source shortest path exploration on the whole graph to count the number of shortest paths existing between the source and all the other nodes in the graph. This operation is extremely power-consuming already on medium-sized networks, and almost prohibitive on large, multi-modal networks, thus jeopardizing the usage of this metric in urban or region-scale scenarios and, particularly, in *dynamic* and *(e.g., travel-time) weighted* settings. In fact, since traffic is highly dynamic and the weighted shortest paths between places in cities change very rapidly especially during rush hours, BC needs to be frequently updated, i.e., recomputed every 15 min in peak times, to have an accurate analysis of the network properties by looking at the spatial distribution of the top-BC nodes (those that attract most of the shortest paths at a given moment of time).

Many algorithms exist to improve the computation of this metric, but, still, the computation time increases quickly with the size of the network, generally

making it impossible to have a real-time computation even in the case of small urban scales. In this paper, we discuss and analyze our original approach, based on machine learning optimisation applied to graph theory, to fasten the computation of BC in large scale networks.

The analyses conducted in this paper pose the bases to design an information system for resilient smart management of large-scale road networks. By relying on such a system, vehicles could be promptly informed about the availability of alternative shortest paths to follow in order to rapidly reach their destination from the current location, thus contributing to globally reduce network congestion and therefore vulnerabilities, by pushing towards a more homogeneous exploitation of the network. At the same time, the information derived from BC computation allows urban planners and managers to perform what we call as *ahead flows monitoring*. With this term, we refer to the ability of our monitoring system to exploit local traffic information to produce a global view of flows distribution that gives insights about possible reorganization of traffic in the near future.

2 State of the Art

The analysis of weighted graphs via BC can provide additional information for studying a urban system and monitoring its evolution [10, 11]. In particular, good correlation exists between traffic flows and node betweenness centrality, especially when the network is congested [8]. Similar conclusions were drawn for a related metric, i.e., dynamic efficiency, on multi-modal networks by Geroliminis et al. [6]. In our previous work [12], it is further observed that edges or nodes identified by higher values of BC, at a certain time, could represent critical regions of the network where traffic will most-likely concentrate in the near future, especially if the areas connected by these nodes are associated to high demand or topological importance.

This property confers great value and, specifically, a predictive power to the BC metric: detecting high-BC nodes can be useful to anticipate were traffic could concentrate in the near future. This aspect paves the way to more informed decision making systems: by continually computing BC to identify the most relevant nodes of a temporal-evolving weighted graph and by analyzing the changes of their distribution in space, dynamic regulation strategies or control actions could be enacted to improve network resilience of the multi-modal transport system. However, recomputing BC with high frequency to follow and even anticipate traffic dynamics, requires extremely rapid and scalable algorithms which exploit network properties to significantly reduce the exploration phase of shortest paths.

Brandes algorithm [13] is a fast algorithm based on explorations of the graph from each node as source and the computation of the contributions during a phase of back propagation. The algorithm has a complexity of $O(nm)$ which is too high for computation in real-time for large networks such as transportation networks.

Different types of strategies have been proposed to reduce the computation time, for example the **approximation** of the BC by reducing the set of shortest paths to explore, which is the strategy we adopt in this paper. Our approach, based on previous work on the topic [14, 15, 5, 16], allows to accurately estimate, with a very low error, nodes BC by clustering the graph to identify representative nodes which will be the sources for the exploration of the graph, the main goal being to reduce the number of explorations through the whole graph, by preserving a very small error in the computation of BC.

3 Contribution and results

Our approach is based on the reduction of the number of explorations in the graph by identifying representative nodes called pivot nodes. The first step to identify pivot nodes is to cluster the network into communities using Louvain clustering. The clustering method aims at maximize the modularity [17]. We compute the BC inside the communities in parallel, a community being considered as a strongly-connected subgraph of G. We call this BC *local BC* as it represents the BC for shortest paths from source to destination of the same community.

Inside each community, we define border nodes as nodes linked to at least one node from an other community. During the *local BC* computation we store the number of shortest paths to reach all the border nodes of a community and the distance, which we normalize. This allows us to group nodes into **classes of equivalence** (i.e. nodes having the same normalized distances and sigmas to reach border nodes). The nodes of a class of equivalence produce the same contributions to nodes outside their community.

We choose one representative node in each class of equivalence, called *pivot*, which will be the only source for the explorations of the graph from all of its class (instead of an exploration from each node of the class). The contributions produced by the pivot are the same than the ones produced by the nodes of the same class of equivalence to nodes outside the cluster of the pivot, but not to nodes inside.

This method is efficient if the number of pivots $k << |V|$ which is not the case for a large (road network) graph. To reduce the number of classes, we perform a second clustering that groups classes into **super classes**, via K-Means. To minimize the error, we choose as pivot the node with the lowest local BC (i.e. the most isolated node). Finally, the contributions are multiplied by the cardinality of the **super class**. The sum of the *local BC* and the total of contributions provides the approximated BC on all nodes.

Algorithm 1 W2C-Fast-BC: algorithm for fast computation of betweenness centrality (pseudo-code of the main function)

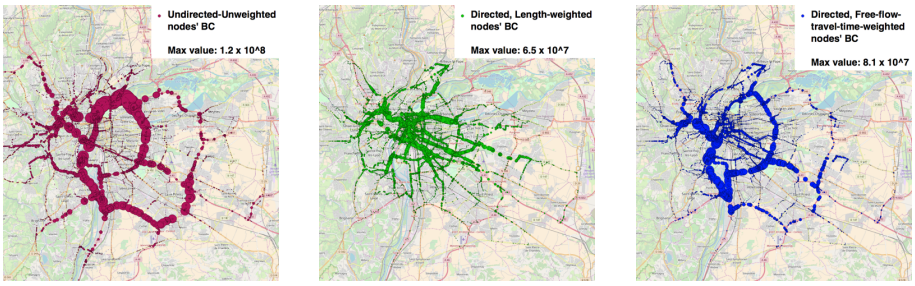
function W2C-FastBC($\hat{\mathbf{G}}, \mathbf{C}, kFrac$)
  $\mathbf{C} \leftarrow weightedLouvainClustering(\hat{\mathbf{G}})$
  $\mathbf{bordernodes_i} \leftarrow findBorderNodes(\hat{\mathbf{G}}, \mathbf{C_i})$
  $\mathbf{local\delta_i} \leftarrow computeLocal\delta(i, \mathbf{C}, \mathbf{bordernodes})$
  $localBC_i \leftarrow local\delta_s(i) + local\delta_z(j)$
  $\mathbf{superClasses_i} \leftarrow WkMeansClustering(\mathbf{C_i}, \mathbf{classes_i}, kFrac)$
  $P_i \leftarrow selectPivotOf(\mathbf{superClasses_i}, \mathbf{localBC})$
  $\delta_\mathbf{i} \leftarrow computeFrom(P_i)$
  $\delta_\mathbf{i} \leftarrow (\delta_\mathbf{i} - \mathbf{localBC}) \cdot |\mathbf{superClasses_i}|$
  $BC_i \leftarrow \delta_s(i) + \delta_z(j)$
  for $i \leftarrow 1, |\mathbf{V}|$ **do**
    $BC_i \leftarrow BC_i + localBC_i$
  end for
  return BC
end function

For more details on the algorithm, the interested reader can refer to our previous work [16]. As the main contribution of this paper, we analyze a very large road network graph, detailed next, in different scenarios to firstly prove the importance of dynamically computing weighted BC. We also exploit a large dataset of GPS observations that let us retrieve periodically changing travel-time weights and build different timestamped snapshots that represent our temporal evolving road-network graph. The analysis confirms the relevance of using BC for traffic prediction and, particularly, the need for a stringent requirement in terms of computation time. This requirement motivates the need for a quick algorithm for BC computation over large, directed and weighted graphs. Secondly, we prove that our solution is capable of satisfying these stringent requirements, thus allowing for (almost) real-time computation of BC.

3.1 Dataset

For our analyses, we rely on the entire road network of the Rhone department, France. The graph, called **Rhone-ROADS** in the following, corresponds to the whole agglomeration of Lyon and its surroundings, with a geographical extent of approximately 3,300 Km^2. This dataset was created using digital maps supplied by the French National Institute of Geographic Information (IGN). The whole network consists of 117,605 nodes and 248,337 edges. In Sec. 3.2, we consider multiple weighted, directed and static graphs that have been derived from the Rhone-ROADS network by selecting some of the available topological attributes (e.g., road segment lengths, capacity, free-flow travel times, etc.) as weight for the edges.

In Sec. 3.3, to evaluate our algorithm for efficient BC computation in realistic dynamic settings, we leverage an additional dataset, namely **Rhone-TAXIS**, used for extracting reliable time-varying traffic information for a portion of the Rhone-ROADS network. The Rhone-TAXIS dataset contains anonymized GPS traces of taxi trips, observed over the Rhone department. The source



(a) Undirected, unweighted (circle size proportional to node's BC)

(b) Directed, length-weighted (circle size proportional to node's BC)

(c) Directed, free-flow-travel-time-weighted (circle size proportional to node's BC)

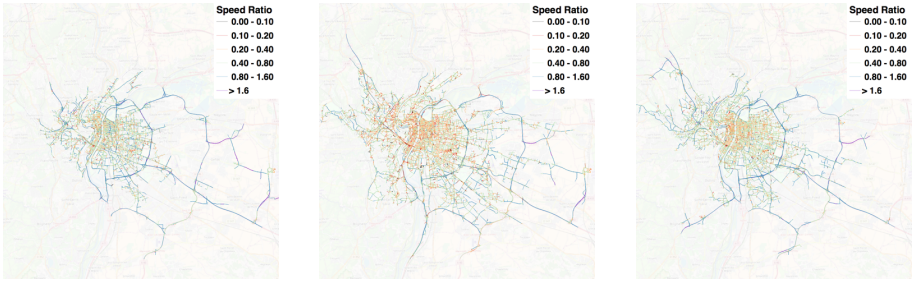
Figure 1: Comparison of BC values in three static instances of Rhone-OBS.

dataset has been collected by the French operator Radio Taxi via a fleet of approximately 400 taxis, during 2011-2012. The dataset logs geo-referenced taxi trips, segmented according to a variable sampling interval (between 10 and 60 seconds), with a global average of 800,000 measurements per day. As an indicator of traffic dynamics, we adopt the median speed observed over each road segment (i.e., an edge of the network) during a fixed-duration of observation (e.g., 1-hour time slots). In order to improve the quality of the Rhone-TAXIS dataset and properly compute our traffic indicator, we have filtered observations with unrealistic speeds (i.e., higher than 130 Km/h). We also use a K-nearest neighbours regression (KNR) [18] as a spatial interpolation technique to retrieve weight information for those links of the Rhone-ROADS network with no or very limited observations, in order to have a full connected, temporal weighted graph.

3.2 BC on Static Graphs

As a preliminary step for the evaluation of BC on *dynamic* graphs, we analyze the Lyon road network in three *static* configurations, i.e.: *i)* undirected and unweighted, *ii)* weighted according to the length of each road segment and directed according to road direction, *iii)* weighted according to Free-Flow Travel Time (FFTT) and directed according to road direction. The objective of this first analysis is not to evaluate the performance of the W2C-Fast-BC algorithm but to shed light on the usefulness of the BC metric with weighted (and directed) graphs, thus proving that different kinds of weights may grasp different notions of vulnerability. Therefore, to simplify the analysis, we consider a sub-graph of the Rhone-ROADS graph, that we call the **Rhone-OBS graph**. A performance analysis is instead presented, in dynamic settings, in Sec. 3.4, by leveraging the largest available weighted dynamic graph, i.e., the Rhone-ROADS KNR-interpolated graph.

The three static configurations of the Rhone-OBS graph are graphically represented in Fig. 1: for each node, we report BC values as circles with diameter



(a) 05:00 (b) 08:00 (c) 20:00

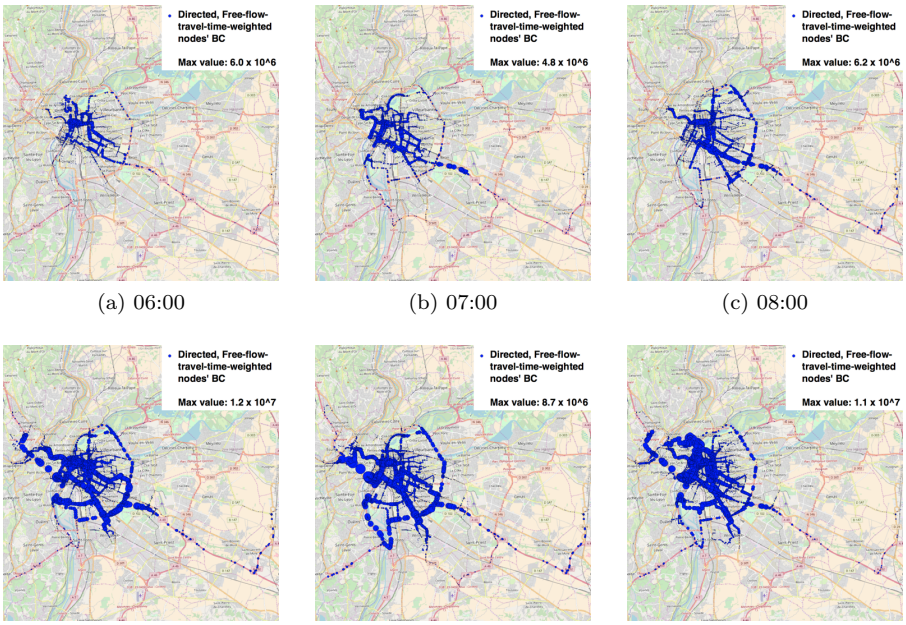
Figure 2: The dynamic taxi graph: median-speed-to-max-speed ratio at different hours of the day. Edge color (red to blue) indicates higher speed-ratio, i.e., reduced congestion)

proportional to BC value. Edges are filtered out for the sake of readability. The visual inspection of the different figures of BC, makes it evident the effect of using different weights for BC computation. Particularly, it can be noticed that on the undirected, unweighted version of the Rhone-OBS graph (Fig. 1a), top-BC nodes are mostly positioned over the city ring road and on top of major highways; urban arterials host the majority of top-BC nodes in the case of the road-length weighted directed graph (Fig. 1b); finally, for the directed, FFTT-weighted instance of the Rhone-OBS graph (Fig. 1c), top-BC nodes can be observed on both arterials and ring roads, with a more homogeneous distribution of BC values with respect to the unweighted, undirected case.

3.3 BC on Dynamic Graphs

The Rhone-OBS graph has been leveraged to extract multiple weighted graph instances, depending on the specific time slot we consider in our analysis. The final temporal weighted graph is composed of 24 snapshots, each corresponding to one hour of the typical day, with different weight configurations derived from the median speed observed on each link in that hour of the day (from the Rhone-TAXI graph). The weight of each edge is the estimated travel time to cross the corresponding road segment, by dividing its length by the estimated median speed. A few instances of this temporal graph are graphically shown in Fig. 2.

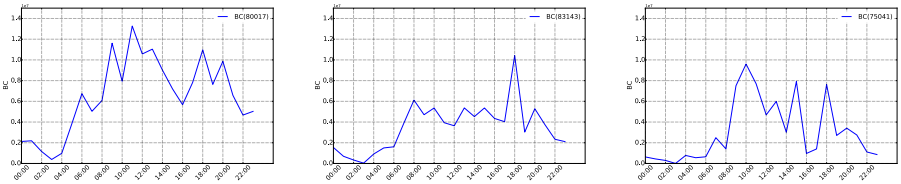
We present in Fig. 3 the spatial distribution at different hours of high-BC nodes, computed on the weighted dynamic graph. The visual inspection of the sub-figures denotes high variability in time and space of the nodes with higher values of travel-time weighted BC (termed TTBC in the following), as also confirmed in Fig. 4, which depicts the evolution over time of TTBC associated to the node with the largest value of TTBC at given time slot. These results provide evidence on the importance of using dynamic, weighted graphs in the computation of BC as well as the need for a rapid algorithm for computation of up-to-date TTBC values. In that sense, the figures unfold interesting dynamics



(a) 06:00 (b) 07:00 (c) 08:00

(d) 09:00 (e) 10:00 (f) 11:00

Figure 3: Nodes' travel-time-weighted BC over the dynamic graph in the time range [05:00 - 10:00]. The size of each circle in the subplots is proportional to node's BC.



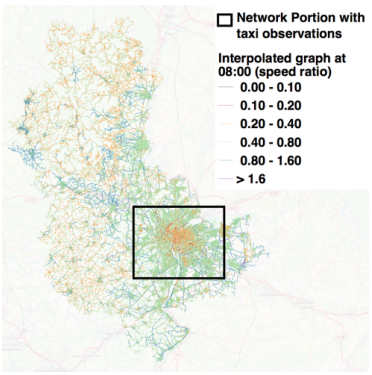
(a) Node with highest BC at 06:00: evolution of its BC over time

(b) Node with highest BC at 08:00: evolution of its BC value over time

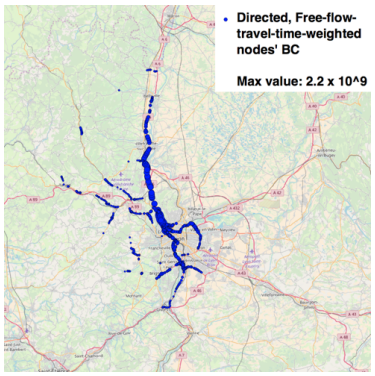
(c) Node with highest BC at 10:00: evolution of its BC value over time

Figure 4: Evolution over time of the top-BC node for some time slots of the dynamic taxi graph.

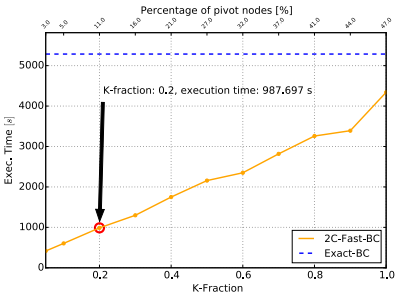
of people's mobility in the city of Lyon: during morning peak hours (Fig. 3c and Fig. 3d), higher TTBC nodes concentrate along the high-speed Lyon ring road, which thus represents one of the most important (and therefore critical) connecting roads of the city, being traversed (in those time slots) by the largest number of the weighted shortest paths. Similarly, some specific arterials traversing the city center also appear as associated with higher TTBC. It is worth noting that



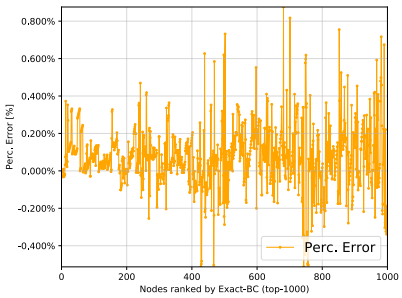
(a) KNR-interpolated graph at 08:00



(b) Top-1000 nodes' BC values at 08:00



(c) Execution time of W2C-Fast-BC vs Brandes-BC at 08:00



(d) KNR-interpolated top-1000 BC percentage error at 08:00

Figure 5: The interpolated dynamic taxi graph: median-speed-to-max-speed ratio at different hours of the day.

such critical roads change frequently, depending on the specific time slot, thus unveiling high traffic dynamics at morning peak hours.

3.4 Performance analysis

In this section, we exploit our W2CFastBC algorithm to perform static and dynamic analysis of the metropolitan road network of the Rhone region, by relying on the largest, KNR-interpolated, temporal graph (see Fig. 5a).

W2CFastBC is written in *Scala* with the *Apache-Spark* framework, leveraging multi-core processing for parallel execution. Spark was configured to work in the standalone cluster mode on two Intel Xeon E5 2640 2.4 GHz multi-core machines, each equipped with 56 virtual cores and 128 GB of DDR4 RAM. All the algorithms used for BC computation leverage 10 cores, by partitioning the map-reduce tasks on two Spark workers, each equipped with 5 executors.

The top-1000 values of BC associated with this graph are reported in Fig. 5b.

As it can be observed from Fig. 5c, the exact algorithm for computing BC on the weighted graph requires a computation time of more than one hour, therefore unable to complete within the duration of the time slot, thus making the computation of BC values at time slot 08:00 completely useless to provide any knowledge (i.e., prediction of traffic flows for the next time slot) that could be exploited to inform travelers about congested roads. Conversely, our W2C-Fast-BC computes in only 987 seconds (i.e., approximately 15 minutes), with a largely tolerable percentage error of 0.8% over the top-1000 BC nodes. Similar results have been observed over the whole dynamic graph, thus proving the adequacy of our solution for quasi real-time monitoring of dynamic, weighted road-networks. The information derived from BC computation along with the related possible reorganization of traffic flows allow urban planners and managers to perform what we call ahead monitoring, since it is possible to estimate where traffic is likely going to concentrate in the near future.

4 Future work

Our approach reduces BC computation time and allows real time monitoring of transportation networks. Since an estimated version of BC could lead to a misunderstanding of traffic conditions, we are currently working on an exact version of the algorithm that will be useful to precisely characterise the boundaries of the error in the approximated approach and to identify effective heuristics for improving the accuracy.

Our future work also targets time-varying computation of BC on multimodal networks to have a more realistic representation of transport dynamics.

References

[1] A. Kazerani, S. Winter, Can betweenness centrality explain traffic flow?, in: 12th AGILE international conference on geographic information science, 2009, pp. 1–9.

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

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

[4] S. Derrible, Network centrality of metro systems, PloS one 7 (7) (2012).

[5] A. Furno, N.-E. El Faouzi, R. Sharma, E. Zimeo, et al., Fast computation of betweenness centrality to locate vulnerabilities in very large road networks, in: Transportation Research Board 97th Annual Meeting, 2018.

[6] L. Bellocchi, V. Latora, N. Geroliminis, A complex networks approach for dynamical efficiency in multimodal transportation systems.

[7] A. Kazerani, S. Winter, Modified betweenness centrality for predicting traffic flow, 2009.

[8] Y. Altshuler, R. Puzis, Y. Elovici, S. Bekhor, Alex, Pentland, Augmented betweenness centrality for mobility prediction in transportation networks, 2012.

[9] S. Zhao, Z. Pengxiang, C. Yunfan, A network centrality measure framework for analyzing urban traffic flow: A case study of wuhan, china 478 (03 2017).

[10] L. Dall'Asta, A. Barrat, M. Barthelemy, A. Vespignani, Vulnerability of weighted networks, Journal of Statistical Mechanics: Theory and Experiment 2006 (04) (2006) P04006.

[11] I. X. Y. Leung, S.-Y. Chan, P. Hui, P. Lio, Intra-city urban network and traffic flow analysis from gps mobility trace, arXiv:1105.5839v1.

[12] E. Henry, L. Bonnetain, A. Furno, N.-E. El Faouzi, E. Zimeo, Spatio-temporal correlations of betweenness centrality and traffic metrics, in: 2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), IEEE, 2019, pp. 1–10.

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

[14] A. Furno, N.-E. El Faouzi, R. Sharma, E. Zimeo, Two-level clustering fast betweenness centrality computation for requirement-driven approximation, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 1289–1294.

[15] A. Furno, N.-E. El Faouzi, R. Sharma, E. Zimeo, Reducing pivots of approximated betweenness computation by hierarchically clustering complex networks, in: International Conference on Complex Networks and their Applications, Springer, 2017, pp. 65–77.

[16] A. Furno, N.-E. El Faouzi, R. Sharma, E. Zimeo, Fast approximated betweenness centrality of directed and weighted graphs, in: International Conference on Complex Networks and their Applications, Springer, 2018, pp. 52–65.

[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) P10008. doi:10.1088/1742-5468/2008/10/p10008.

[18] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.