An Improved Method for Discovering Link Criticality in Transport Networks

Juan Segovia, Eusebi Calle, Pere Vilà
Institute of Informatics and Applications (IIiA), University of Girona,
Girona 17071, Spain
Email: {jsegovia, eusebi, perev}@eia.udg.edu

***Abstract*—Quantitatively assessing the importance or criticality of each link in a network is of practical value to operators, as that can help them to increase the network's resilience, provide more efficient services, or improve some other aspect of the service. Betweenness is a graph-theoretical measure of centrality that can be applied to communication networks to evaluate link importance. However, as we illustrate in this paper, the basic definition of betweenness centrality produces inaccurate estimations as it does not take into account some aspects relevant to networking, such as the heterogeneity in link capacity or the difference between node-pairs in their contribution to the total traffic. A new algorithm for discovering link centrality in transport networks is proposed in this paper. It requires only static or semi-static network and topology attributes, and yet produces estimations of good accuracy, as verified through extensive simulations. Its potential value is demonstrated by an example application. In the example, the simple shortest-path routing algorithm is improved in such a way that it outperforms other more advanced algorithms in terms of blocking ratio.**

***Index Terms*—Routing in GMPLS networks, link criticality, betweenness centrality.**

I. Introduction

In data communication networks, the number, properties and usage of links have direct impact on the capabilities of the network and the quality of the services they can provide. This is even more so in transport networks, where the service takes the form of virtual path(s) between two of its nodes. This service is called "connection", and the resources assigned to it, e.g., capacity, remain allocated for the duration of the service. Depending on the properties of the network and the policies applied by its operator, links play a different role. Some of them can be central in the sense that a large proportion of traffic passes through them, while others might be on the periphery in this usage category. This differentiation of roles happens independently of the physical location, or the nodal degree of their end-nodes.

Understanding and assessing the importance or centrality, we might even say "criticality", of each link is of great value. For example, a network operator whose service level agreement includes performance guarantees under failure, might use that knowledge to decide upon which protection scheme (such as 1+1, 1:1 and 1:N; see [1]) best guards against specific failure events. Likewise, it can be used for scheduling maintenance activities, as well as for appraising the merits of capacity or equipment upgrades proposed to ease or avoid congestion, increase resilience, or improve some other aspect relevant to a service. If information about node centrality is added, it can also help, for example, in planning or evaluating the adequacy of communication or computational resources in the control plane of a GMPLS-based network. Certainly, it can also be used to study how catastrophic events such as earthquakes and flooding can affect the network, as exemplified by the work in [2].

Betweenness centrality is a graph-theoretical concept that measures the degree to which a vertex acts as an intermediary in the communication between every pair of vertices in its graph [3]. It has been used to study aspects and properties of complex networks, be it those representing personal communication and relationship between people, power transmission grids or data communication networks such as the Internet.

Just as with vertices, edge betweenness centrality can be defined for the same purpose: understanding its centrality. However, in this paper we reason about why the basic definition of edge centrality does not correctly identify the true link centrality in a transport network, and propose a new algorithm, which we call the *Capacity- and Traffic-aware edge betweenness centrality algorithm*, or CTA. A salient attribute of this algorithm that its input are static or slowly-changing attributes of the topology, but predicts with good accuracy the effective usage of links in a dynamic traffic scenario. Moreover, it has essentially the computationally complexity as the basic betweenness centrality.

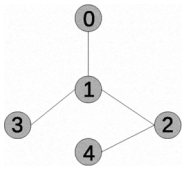
The remainder of the paper is organised as follows: In section II, the formal definition of betweenness centrality is given, together with an illustrative numerical example. In section III, the shortcomings of the basic edge betweenness definition for data communication networks are highlighted, and our new algorithm is presented. Section IV explains the result of the verification carried out through simulation, and section V shows an application of the new algorithm to improve shortest-path routing. Finally, the section VI concludes the paper.

II. The concept of betweenness centrality

Betweenness centrality can be defined and computed in several ways, depending on the application. The one of interest for our study is the so-called geodesic betweenness [3], which determines how often a node of a given network topology lies

This work was partially supported by the Spanish Science and Technology Ministry (MCYT) research project TIC2003-05567 and by the research support program of the Generalitat of Catalonia (SGR 00296).

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266



Node	$C_B(v)$	Link	$C_B(e)$
0	0.4	0–1	0.4
1	0.9	1–2	0.6
2	0.7	2–4	0.4
3	0.4	3–1	0.4
4	0.4		

Fig. 1. Example of node (vertex) and link (edge) betweenness centrality

along the shortest path between all the possible pair of nodes. More formally, it is usually defined as follows:

Denote by $\sigma(s,t)$ the number of shortest paths that exist between nodes s and t, and by $\sigma(s,t|v)$ the number of shortest paths between nodes s and t passing through a given node v other than s and t. E is the set of edges in the graph G, and V the set of nodes. Then, the shortest path betweenness centrality $C_B(v)$ for some node v is

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \quad (1)$$

However, as our interest is in measuring the centrality of links and not nodes, we need the *edge betweenness centrality* $C_B(e)$ instead. This is achieved by replacing $v \in V$ with $e \in E$ in (1), where e is any given link in the topology. For practical reasons, the obtained betweenness centrality is usually normalised.

In the original conception of betweenness centrality, "shortest path" is defined in terms of the number of edges traversed, that is, the number of hops. This assumption permits the use of a simple breadth-first search algorithm to identify the shortest paths. However, when the cost can be an arbitrary value, such as the link's physical length in kilometres, this approach is no longer possible, requiring more complex path exploration algorithms. Brandes [4] gives several algorithms for the efficient computation of betweenness centrality. The one for weighted graphs employs Dijkstra, backed by a priority queue. Its running time is bound by $O(|V||E|+|V|^2 log\ |V|)$.

Fig. 1 gives an example of node and link betweenness centrality in a small network topology. The graph is depicted as undirected, although the computation is performed on the equivalent directed graph. As expected, $C_B(1)$ is the highest, given the central position of node 1 in the topology, followed by $C_B(2)$. Likewise, the higher relative importance of link 1–2 is also correctly reflected (0.6 versus 0.4); of the twenty shortest paths that exist, twelve pass through it (six in each direction), instead of eight, as with the rest.

III. An Improved Edge Betweenness Centrality Algorithm

In this section we highlight two important shortcomings of the existing edge betweenness centrality algorithm when applied to path-oriented networks (e.g. G/MPLS), and propose two modifications to improve it.

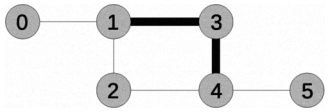


Fig. 2. A topology with non-homogeneous link capacities

A. Shortcomings of the basic edge betweenness centrality algorithm

Our previous example is useful to illustrate the concept of betweenness centrality, but we need a more realistic one. In the topology of Fig. 1, connectivity is too low (there is only one possible path for each node pair), and we discarded the links' capacity by implicitly assuming that they are uniform across the board. Fig. 2 shows a more interesting topology. Let's assume that the links depicted with thick lines have double the capacity of the thin ones. Note that the nodes 1, 2, 3 and 4 form such a structure that several paths of equal length exist. For this topology, it would not be wise to treat links 1–3 and 1–2 as equally central because more paths are expected to traverse link 1–3 than link 1–2 during the network operation, simply because there will be more free capacity in the former. Therefore, $C_B(1\text{–}3)$ should be higher than $C_B(1\text{–}2)$ to reflect this fact. However, the original betweenness centrality algorithm computes the $C_B(e)$ in such a way that multiple alternate paths evenly share their centrality (the centrality value is divided). This approach can also be found in the algorithm that serves as the basis for all the variants given in [4]. Certainly, it is possible to devise a scheme in which the alternate paths have a different contribution weight, for example based on their maximum capacity. However, such paths are not necessarily disjoint. In addition, in path-oriented data networks, the traffic of a given connection is, in general, not split into separate subflows towards its destination. These considerations highlight the complexity of finding the most appropriate weighing scheme. We describe now a straightforward approximating approach that we have found to work well in practice.

B. The CTA edge betweenness centrality algorithm

Firstly, we propose replacing the breadth-first shortest path search, together with the assumption about shared contribution, by one that simultaneously takes into account the distance (in hops) from the origin towards the destination and the largest minimum capacity along the feasible path, as follows:

First modification: Should two paths of equal hop count exist, the one with more capacity is preferable. In other words, apply a widest-shortest path algorithm (WSP) [5] to identify the (preferred) path between the end nodes. Table I illustrates the positive effect that this change has on the computed betweenness centrality. For comparison, the results of both algorithms (the original and the new one we are outlining) are shown. As can be seen, links 1–3 and 3–4 are correctly identified as the two more central elements in the respective topology.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266

TABLE I
EDGE BETWEENNESS CENTRALITY: BREADTH-FIRST SEARCH VERSUS WSP-BASED. SEE TOPOLOGY IN FIG. 2

Link	Edge betweenness centrality	
	BFS (original)	New approach
0–1	0.333	0.333
1–2	0.467	0.200
1–3	0.200	0.467
2–4	0.400	0.133
3–4	0.133	0.400
4–5	0.333	0.333

Another aspect that affects betweenness centrality in a data network is the contribution of each node pair to the total traffic. Therefore, we should consider how to take into account the traffic matrix, if such information is available. The original definition of betweenness centrality implicitly assumes that the contribution is even among all the node pairs, thus the expression "*the number of paths* passing through a given node v" in the description of (1). To account for the potential disparity in the traffic matrix, we propose introducing another modification to the edge betweenness centrality algorithm as follows:

Second modification: Whenever the given edge e appears in a path between the nodes s and t, instead of counting this occurrence as one, consider an increment proportional to the contribution of the node pair (s,t) to the total traffic. To illustrate this, let T be the traffic matrix for the topology of Fig. 2 such that $T(0,5) = 10$, $T(3,5) = 2$, and $\sum_{s,t \in V} T_{s,t} = 100$. The contribution of the node pair $(0,5)$ in the centrality of the edge 4–5 is 0.1, whereas that of the node pair $(3,5)$ is 0.02. In fact, the unavailability of a traffic matrix can be handled as a special case in which $T_{s,t} = 1 \quad \forall s,t \in V, s \neq t$, and $T_{s,t} = 0$ when $s = t$.

Our new capacity- and traffic-aware edge betweenness centrality algorithm (or CTA, for short) is given in full in CTAEDGEBETWEENNESS, which we briefly describe below. The output is a measure of the centrality or criticality of each and every link in the topology, given the links' capacity and, optionally, the traffic matrix.

The main body of the algorithm starts in line 6. For each possible source node, the procedure FINDPATHSCTA is invoked to compute the shortest widest path to every other node (lines 6–7). $\pi[v]$ contains node v's parent node towards the source s. Therefore we can now identify the edges forming the full path from s to every destination t, and appropriately register in σ their participation in these shortest paths (lines 9–15). As σ is used as an accumulator, its elements are initially set to 0 (lines 3–5).

With respect to FINDPATHSCTA, it is worth mentioning that $M[u]$ contains, during path exploration, the largest minimum capacity found along the shortest path leading to a node u. Thus, a node v adjacent to u can be updated to point to u either because passing through u is shorter (in hop count) or there is no difference but the edge u–v has more capacity, information available in C (lines 18–22). In line 23, we use

Algorithm 1 CTAEDGEBETWEENNESS

1: **Input:** Directed graph $G(V,E)$, and traffic matrix T.
2: **Output:** Normalised edge betweenness centrality $\forall e \in E$.
3: **for all** $e \in E$ **do**
4: $\quad \sigma[e] \leftarrow 0$
5: **end for**
6: **for all** $s \in V$ **do**
7: $\quad \pi \leftarrow$ FindPathsCTA(s)
8: $\quad$ **for all** $w \in V$ **do**
9: $\quad\quad t \leftarrow w$
10: $\quad\quad$ **while** $\pi[t] \neq$ **nil do**
11: $\quad\quad\quad e \leftarrow$ edge $(\pi[t], t)$
12: $\quad\quad\quad \sigma[e] \leftarrow \sigma[e] + T_{st}$
13: $\quad\quad\quad t \leftarrow \pi[t]$
14: $\quad\quad$ **end while**
15: $\quad$ **end for**
16: $\quad C_B \leftarrow \sigma$ normalised
17: **end for**
18: **return** C_B

Algorithm 2 FINDPATHSCTA

1: **Input:** Source node s, and links' capacity C
2: **Objective:** Find the shortest paths from s to all the other nodes. Take into account both hop count and largest minimum link capacity.
3: **Output:** $\forall v \in V$, the parent node towards s.
4: $Q \leftarrow$ new priority queue
5: $\pi \leftarrow$ new map (vertex $\rightarrow$ parent vertex)
6: $D \leftarrow$ new map (vertex $\rightarrow$ hop count)
7: $M \leftarrow$ new map (vertex $\rightarrow$ largest min capacity)
8: **for all** $v \in V$ **do**
9: $\quad M[v] \leftarrow \infty$
10: $\quad D[v] \leftarrow \infty$
11: $\quad \pi[v] \leftarrow$ **nil**
12: **end for**
13: $D[s] \leftarrow 0$
14: insert s into Q with priority 0
15: **while** Q not empty **do**
16: $\quad u \leftarrow$ removeMin(Q)
17: $\quad$ **for all** v adjacent to u, v not yet visited **do**
18: $\quad\quad \gamma \leftarrow min(C_{uv}, M[u])$
19: $\quad\quad$ **if** $(D[u] + 1 < D[v])$ **or** $(D[u] + 1 = D[v]$ **and** $\gamma > M[v])$ **then**
20: $\quad\quad\quad M[v] \leftarrow \gamma$
21: $\quad\quad\quad D[v] \leftarrow D[u] + 1$
22: $\quad\quad\quad \pi[v] \leftarrow u$
23: $\quad\quad\quad$ insert v in Q with priority $(D[v] + 1/C_{uv})$
24: $\quad\quad$ **end if**
25: $\quad$ **end for**
26: **end while**
27: **return** π

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266

$D[v]+1/C_{uv}$ as the priority expression for the priority queue. This expression implements the policy of using link capacity to break any tie on path length alone: if two nodes are at the same distance from the source, choose to process next the one to which a link with higher capacity arrives.

After finishing the main loop in CTAEDGEBETWEENNESS (lines 6–17), the values in σ can be normalised to finally become the edge's betweenness centrality. The running time of the algorithm is dominated by that of FINDPATHSCTA, which is in $O(|E| + |V|\ log\ |V|)$ and is repeated $|V|$ times. After that, lines 8–15 traverse the path from the current root node to every other node. The number of iterations of the inner loop (lines 10–14) depends on the length of paths, which in realistic topologies is significantly smaller than $|V|$ and $|E|$. Therefore, the total running time of CTAEDGEBETWEENNESS is in $O(|V||E| + |V|^2\ log\ |V|)$.

IV. VERIFICATION OF THE CTA ALGORITHM

In this section, we describe the steps performed to verify the effectiveness of the CTA algorithm to estimate the importance of links. Our aim is to observe, through simulation, the behaviour of each link in terms of usage, and compare it with the estimation. Therefore, we use the simulation to rank each link of a given topology according to its effective participation in paths, i.e., the number of times it is included in a path by a capacity-constrained shortest-path first routing algorithm in a scenario of dynamic traffic. Once the simulation completes, the edges are arranged by their observed frequency of use in descending order. By "use" we mean the number of times it is selected by the routing algorithm, not the link capacity used. Clearly, this process finds out empirically the edges' centrality, although through a slightly different procedure than before. Consequently, we can compare the centrality statically predicted by the CTA algorithm with the one effectively observed during the simulated routing.

To perform the ranking of edges, we developed an event-based simulator that reproduces the process of route selection in a path-oriented transport network. Thus, it handles the reception of connection requests between node pairs, triggers and coordinates the proper routing and capacity allocation based on the demand, keeps track of the usage of resources (the free capacity on each link), releases connections when their holding time has expired, and collects statistical data.

In the following subsections, more details about the simulation environment are given, as well as the results obtained.

A. Simulation parameters

To subject the CTA algorithm to diverse scenarios, we chose three well-known topologies (see Fig. 3): NSFnet14, KL, and Cost266. Additional variation is introduced by using different traffic patterns and link capacities. The relevant properties of the selected topologies are summarised in Table II.

For path selection, a Dijkstra-based shortest path is employed, where the cost is hop count, i.e., a minimum-hop routing algorithm (MHA). Links that do not have enough free capacity to satisfy the arriving demand are filtered out before the exploration. The decision to use MHA instead of another more intelligent routing algorithms available owes to the prevalence of shortest-path routing in the networks deployed in the real-world, e.g., the Internet.

For each of the topologies under study, we run the simulation under the following traffic and capacity scenarios:

- **Scenario 1**: Homogeneous link capacity (all links set to the same capacity) with uniform traffic matrix (the contribution of traffic of every possible node pair is approximately the same). This is the scenario assumed by the betweenness definition of section II.
- **Scenario 2**: Homogeneous link capacity with non-uniform traffic matrix (node pairs contribute differently to the total traffic)
- **Scenario 3**: A subset of the links have more capacity, but traffic matrix is uniform
- **Scenario 4**: A subset of the links have more capacity, and the traffic matrix is non-uniform.

Each run of the simulator uses a randomly generated input set consisting of a series of connection requests between a pair of nodes in the topology. A connection demands a certain capacity which, if accepted, is allocated for the duration of the service. The source and destination nodes are randomly selected. Capacity is demanded in an abstract unit of traffic in the range 1–10 (randomly selected; uniform distribution). Arrivals follow a Poisson process, with exponentially distributed connection holding times. Link capacities are adjusted so as to reject approximately 1% of the connection requests.

B. Simulation results and discussion

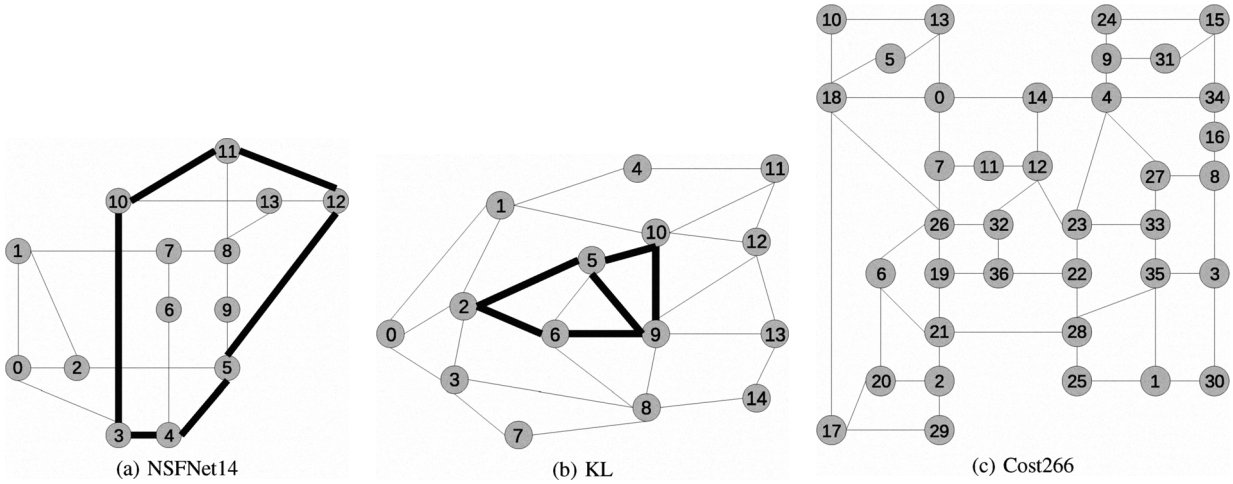
The results reported are the average of 100 runs, each one consisting of 30000 connection requests. After the series of runs for a given scenario had completed, links are ranked according to their frequency of use.

Table III shows the results for the NSFnet14 topology. There we can compare the outcome of the basic algorithm with that of CTA. These subtables shows only the first eight most central links, as selected by each algorithm. The full list is given in graphical form later on. The values in the column $C_B(e)$ are sorted from highest edge betweenness centrality to lowest, thus giving an estimated ranking that can be contrasted with the product of the simulation. According to Table III(a), for example, the most frequently used link during routing should be 8–7, followed by 10–3, and so on. Recall that this estimation is independent of the simulation as it is purely based on static properties.

TABLE II
MAIN PROPERTIES OF THE TOPOLOGIES USED IN THE SIMULATIONS

Property	NSFnet14	KL	Cost266
Number of nodes	14	15	37
Number of edges	21	28	57
Avg. nodal degree	3.00	3.73	3.13
Network diameter	6 hops	5 hops	11 hops

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266



(a) NSFNet14 (b) KL (c) Cost266

Fig. 3. Topologies used in the simulation

TABLE III
ESTIMATED VERSUS OBSERVED EDGE CENTRALITY OF EIGHT LINKS IN NSFNET14. LINK CAPACITY: HOMOGENEOUS, TRAFFIC MATRIX: NON-UNIFORM (SCENARIO 2)

Link	Estimation		Observed link usage ranking							
	$C_B(e)$	Rank	1	2	3	4	5	6	7	8
8–7	0.1648	1			100					
10–3	0.1539	2								100
5–2	0.1429	3		41	59					
12–5	0.1319	4	100							
11–8	0.1319	5								
3–0	0.1319	6		62	38					
9–5	0.1209	7								
5–4	0.1209	8								

(a) With the basic edge betweenness algorithm

Link	Estimation		Observed link usage ranking							
	$C_B(e)$	Rank	1	2	3	4	5	6	7	8
12–5	0.0794	1	100							
5–2	0.0707	2		41	59					
3–0	0.0698	3		62	38					
8–7	0.0650	4				100				
1–0	0.0630	5					97	3		
13–12	0.0572	6						6	94	
2–0	0.0550	7					3	91	6	
10–3	0.0537	8								100

(b) With the CTA algorithm

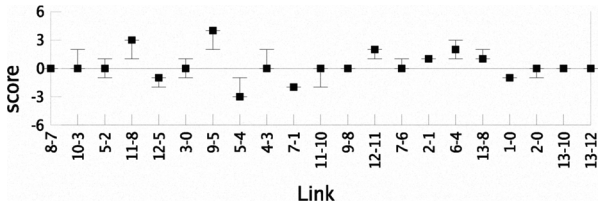
The results of the simulation are grouped under the column "Observed link usage ranking". Table III(a) shows that, in the 100 runs, the link 12–5 turned out to always occupy the first position, i.e., it was always the most used link (although the basic betweenness centrality algorithm estimated that it would be in the fourth position). The link 5–2 was expected to be the third most used one, but in 41 runs out of 100, it was the second, and the third in 59 of the runs.

Now we are in a position to compare and contrast the results of both algorithms (basic versus CTA): Firstly, the links selected as well as their ordering are different, though some overlapping exists. Secondly, the expected and observed rankings do not match in Table III(a), to the point that the results for several links (1–8, 9–5, 5–4) are not in the displayed region of the table, i.e., they are beyond the eighth column. On the other hand, in Table III(b) the values are arranged so that a trend is clearly visible in the form of a "diagonal line", indicating a good match between the expected and observed rankings. Note that the frequency of use of some of the links are spread over different columns (see links 5–2 and 3–0, for example), and one of the links (2–0) is slightly off its expected position (expected in the 7th; appeared mainly in the 6th). Certainly, such spreading is expected, given the randomness in the data set and the workings of the routing algorithm, which has to take into account both residual capacity and path length.

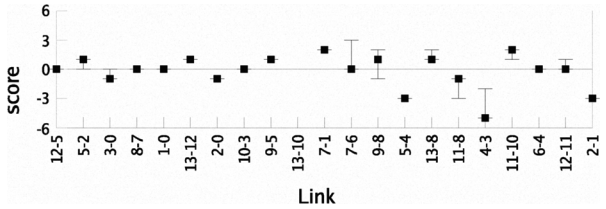
The values in Table III(b) give a sense of the suitability of the CTA algorithm to estimate the importance of links in the NSFNet14 topology in the scenario 2. In a more compact format, Fig. 4–6 present the results of the simulation and the comparison for the other scenarios and network topologies, with the added advantage of the inclusion of all links, not just the first eight. In these figures, links appear on the x-axis from left to right according to their ranking of centrality as estimated by the CTA algorithm. A score that captures the divergence (or closeness) between the observed and the estimated ranking is assigned to each link and plotted on the y-axis. This score is the algebraic difference between the link's most frequent observed position during the simulation (i.e., the modal value of the frequencies of use) and its estimated ranking. Therefore, a perfect link-by-link match would yield a straight line (i.e., $y(e)_{e\in E} = 0$). As the data points refer only to the modal value, we use error bars to give a hint of how much spreading is involved.

Fig. 4 is for the NSFnet14 topology in the four scenarios. This figure confirms, firstly, that the CTA algorithm produces different rankings for each scenario. Secondly, that the correlation between the expected and observed ranking is good; the lowest coefficient of determination r^2 is 0.85. Visually, the best fit corresponds to scenario 4, disregarding its two outliers (links 13–10 and 9–8). In fact, the observed ranking of these two links are exactly in reverse order with respect to

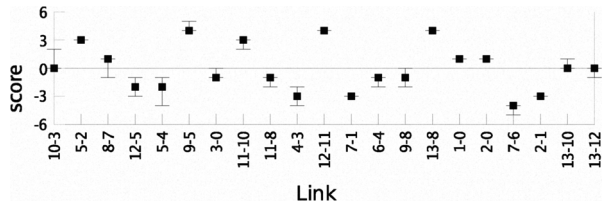
Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266



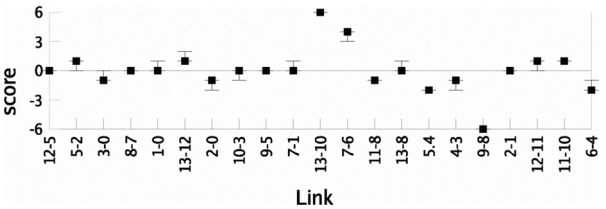
(a) Homogeneous link cap., uniform traffic (scenario 1). $r^2 = 0.93$



(b) Homogeneous link cap., non-uniform traffic (scenario 2). $r^2 = 0.86$



(c) Non-homogeneous cap., uniform traffic (scenario 3). $r^2 = 0.85$

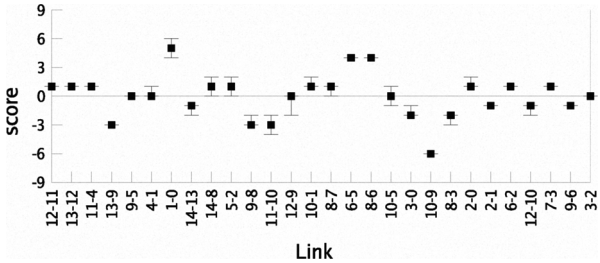


(d) Non-homogeneous cap., non-uniform traffic (scenario 4). $r^2 = 0.87$

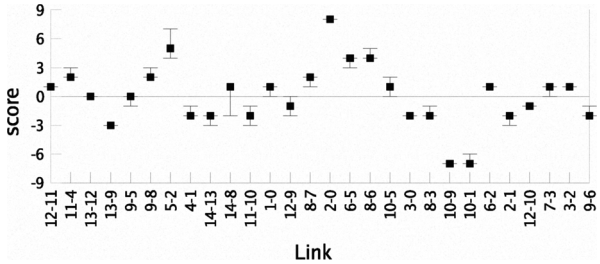
Fig. 4. Estimated versus observed centrality. Topology: NSFnet14

the estimation; otherwise, the r^2 would be 0.96.

The results for the KL topology in the scenarios 2 and 4 are presented in Fig. 5. As can be seen, they are similar to the ones obtained for the NSFNet14 topology. The absolute value of the largest mismatch is higher (i.e., the distance from 0) but the number of links in KL is also higher, such that the maximum relative distance are in fact similar in both cases. As for the most complex topology considered in this study, Cost266, Fig. 5 shows the result for scenario 2. We can see that the mismatch between observed and estimated ranking grows larger than in the other topologies as we move towards the middle of the x-axis, and the spreading is also larger. However, this not different than the other cases in relative terms because this topology is almost three times larger in number of links than the others (57 versus 21 and 28). The coefficient of determination r^2 is 0.91, equally good as the previous ones. Note that the data points for links 23–12 and 33–23 are not plotted (their scores are -17 and -14, respectively), but they are properly accounted for in r^2. The results for the other



(a) Homogeneous link cap., non-uniform traffic (scenario 2). $r^2 = 0.92$



(b) Non-homogeneous cap., non-uniform traffic (scenario 4). $r^2 = 0.86$

Fig. 5. Estimated versus observed centrality. Topology: KL

scenarios are similar, thus we omit further details.

Summarising, these results show that the CTA algorithm is effective in identifying the centrality or importance of each link in a given topology in relation to the rest, even when they differ in their capacity or when the traffic contribution of node pairs is uneven, situations under which the basic edge betweenness centrality algorithm fails. The input to the CTA algorithm is minimal and static, and readily available for a given network without performing any actual routing, and yet it manages to estimate with a high degree of accuracy the link usage dynamics in the scenarios considered in this paper, as the simulations show.

V. Example of application: CTA and shortest-path routing

One well-known drawback of the shortest-path routing algorithm (SPA) is that it tends to quickly overuse certain resources by repeatedly selecting the same subset of links until saturation, and only then switching to other paths. This provokes the rejection of demands that could have been accepted should such a greedy approach had not been used. The greediness nature of the SPA comes from the fact that the path with the least amount of aggregated cost is selected among all the admissible ones. Despite this drawback, the SPA is appreciated for its simplicity, ease of implementation, and relatively low computational requirements.

A. Approaches to overcome the drawbacks of Shortest-Path routing

Two improvements over the basic SPA are Widest-Shortest-Path (WSP) [5] and Shortest-Widest-Path (SWP) [6]. WSP chooses a shortest path of minimum cost, favouring that with the largest residual capacity if more than one exists. Therefore,

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266

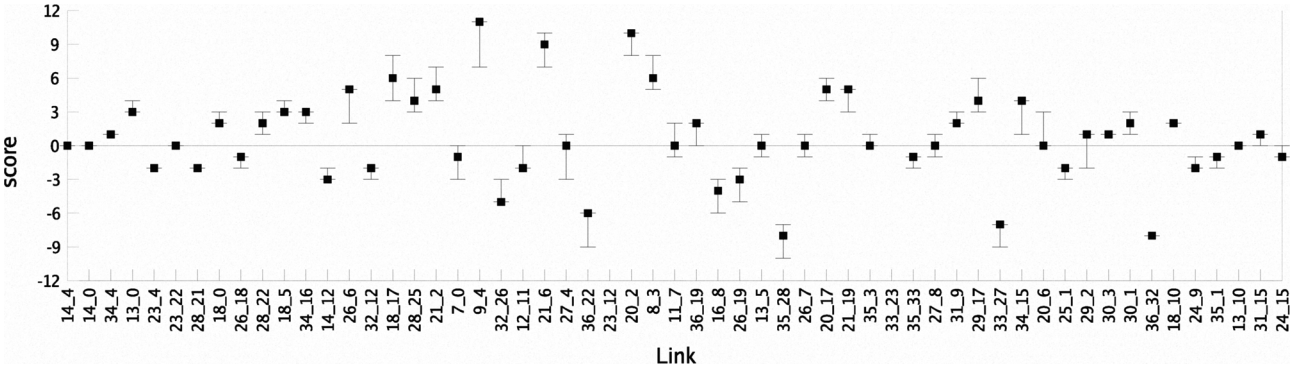


Fig. 6. Topology: Cost266. Homogeneous link cap., non-uniform traffic (scenario 2). $r^2 = 0.91$

it promotes better load distribution than SPA by avoiding links that are already loaded. On the contrary, SWP first finds one or more paths with the largest residual capacity, and then selects the shortest one. Both WSP and SWP are certainly improvements over the basic SPA, but the essential drawback continues there since paths are repeatedly selected until saturation [7].

A totally different approach to path selection is offered by the Minimum-Interference Routing Algorithm or MIRA, introduced in [8]. MIRA aims to route an incoming connection over a path that does not "interfere too much" with some other path that may be critical to satisfy a future demand (demands are required to belong to a predefined set of ingress–egress node pairs). To achieve its objective, MIRA introduces the notion of critical links, which are those with the property that, whenever a connection is routed over them, the maxflow values of one or more ingress–egress pairs decreases. MIRA requires the computation of several expensive minimum max-flows for each connection request, which makes it impractical. Several derivatives have been introduced that try to lower the computational complexity, such as *Light* MIRA (LMIR) [9] and *Simple* MIRA (SMIRA) [10].

The objective of improving the blocking ratio is certainly achieved by MIRA and its derivatives, but their runtime performance is several times worse than that of SPA, even in simplified versions such as LMIR. Moreover, contrary to SPA, MIRA has the requirement that each routing node have an up-to-date view of the residual capacity in the network (a requirement also shared with WSP/SWP).

In light of these considerations, improvements to the shortest-path routing algorithm are desirable and of practical importance. We show in the following subsection preliminary results of the use of the CTA algorithm to influence and improve the performance of the shortest-path routing algorithm.

B. Improving the simple Shortest-Path routing with CTA

As discussed above, the problem with SPA is that it builds bottlenecks in the network by saturating certain links. Given that our CTA edge betweenness algorithm gives an estimation of link usage, we can use this information to rebalance the link's importance. Such rebalancing can be obtained by defining a cost function that makes less attractive (more costly) those that are initially identified as very central (i.e., very prone to saturation), and increasingly more attractive those that the CTA algorithm identifies as less important.

Let C_{BR} be the edge betweenness centrality as produced by the CTAEDGEBETWEENNESS algorithm, sorted from most central to less central, so that $C_{BR}(i)$ is the i-th element in the ranking. Then, we can assign a new cost to each link $w(i)$ with (2):

$$w(i) = \alpha(1 - C_{BR}(i)) \tag{2}$$

where α is an adjustment factor that decreases as we proceed down the list in C_{BR}. For the results reported in this subsection, we used $\alpha = 1.001^k$, where $k = 0, -5, -10, \cdots$.

With this heuristic in place, a simple capacity-constrained minimum-cost shortest path routing can be used (i.e., a weighted Dijkstra). For convenience, let's call this algorithm the SP_CTA algorithm for "shortest-path CTA-based algorithm". SP_CTA comprises both the initial rebalancing weight assignment and the subsequents Dijkstra-based path computations.

Through simulation, we compare the blocking ratio obtained by SP_CTA, WSP and MHA (minimum-hop shortest-path) with the topologies KL and NSFnet14. The simulation parameters are essentially the same as described in IV-A, the difference being the routing algorithm employed. Three target blocking ratios (1%, 5% and 10%) are defined, the baseline being the number of rejections produced by MHA. To reach each target, link capacities are adjusted by a factor found experimentally. For this evaluation, we used a non-uniform traffic matrix with homogeneous link capacity. The demand set is randomly generated for each run, as described in IV-A, and the three routing algorithms receive that very same input so as to make the comparison fair.

In table IV, the column "Reject." is the number of connections rejected (averaged over 10 runs), and the column "%" is for comparison with the baseline. At the 1% target, SP_CTA attains an improvement of more than 25% over MHA (26% for KL and 29% for NSFnet14), and also outperforms WSP. Fig. 7 illustrates that this happens at the three targets,

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266

TABLE IV
COMPARISON OF THE BLOCKING RATIO OF SP_CTA, MHA AND WSP. THE BASELINE IS MHA AT THE GIVEN TARGET BLOCKING RATIO (MHA_B).

		KL		NSFnet14	
MHA_B	Algorithm	Reject.	%	Reject.	%
1%	MHA	263	100	236	100
	WSP	214	81	193	82
	SP_CTA	195	74	168	71
5%	MHA	1510	100	1487	100
	WSP	1449	96	1326	89
	SP_CTA	1372	91	1325	89
10%	MHA	3023	100	2810	100
	WSP	2951	98	2727	97
	SP_CTA	2844	94	2632	94

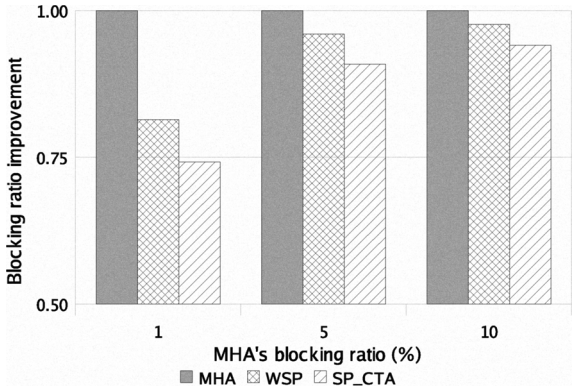


Fig. 7. Connections blocked: WSP and SP_CTA vs MHA. Topology: KL

although the improvement diminishes with the unrealistically high blocking ratios of 5% and 10%. Initially, we also included LMIR in the comparison, but we observed that its blocking ratio was much worse than WSP and SP_CTA. Although further analysis is needed, a possible explanation for this is that LMIR (and MIRA) tries to favour the distinguished set of ingress–egress node pairs, but in our scenarios we do not impose such restriction, and each and every node can act as source and destination.

It is certainly possible to devise an algorithm that directly incorporates the computation of betweenness centrality, acting upon, say, the residual capacity, as in [11]. However, such approach have the drawback of requiring a constantly up-to-day view of the network. It would also increase the computational cost of the algorithm, for computing the betweenness centrality only once is negligible, but doing it repeatedly is not. Therefore, we think that the approach outlined here is interesting because the obtained improvement is the result of the pre-computation step (weight assignment), and not of a change in the routing algorithm itself, which remains one of the simplest available.

VI. CONCLUSION

We have presented in this paper a new algorithm for estimating the importance or criticality of links in a network, which we called the Capacity- and Traffic-aware edge betweenness centrality algorithm, or CTA. This algorithm uses as input only static or slowly-changing attributes of the topology. Contrary to the basic graph-theoretical edge betweenness centrality definition, which is not suitable for real-world data communication networks, as discussed and exemplified in the paper, CTA is able to estimate with good accuracy the importance of each link in relation to the others. Its effectiveness was verified through simulations, which covered several network scenarios. There are many potential applications for this algorithm. For example, it can be used to improve network resiliency by identifying which links are more valuable and hence need more protection, or to assess which sections of the network would require capacity upgrade, among others. As a concrete example of application, the paper presented a variation of the minimum-cost shortest-path routing algorithm that uses CTA's output to better load distribution and thus reduce blocking. The simulations showed that this approach produces lower blocking ratio than widest-shortest-path (WSP) while retaining all the benefits of the simple shortest-path algorithm.

REFERENCES

[1] A. Haider and R. Harris, "Recovery techniques in next generation networks," vol. 9, no. 3, 2007, pp. 2–17.

[2] Z. Huang, M. Woolf, and R. Mondragon, "Building catastrophes: Networks designed to fail by avalanche-like breakdown," no. 9, 2007.

[3] L. C. Freeman, "A set of measures of centrality based upon betweenness," vol. 40, no. 1, 1977, pp. 35–41.

[4] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," vol. 30, no. 2, 2008, pp. 136–145.

[5] R. A. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," 1997, pp. 1903–1908.

[6] Z. Wang and J. Crowcroft, "Quality of service routing for supporting multimedia applications," vol. 14, 1996, pp. 1228–1234.

[7] A. Capone, L. Fratta, and F. Martignon, "Dynamic routing of bandwidth guaranteed connections," 2003, pp. 75–86.

[8] M. S. Kodialam and T. V. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," 2000, pp. 884–893.

[9] G. B. Figueiredo and N. L. S. D. Fonseca, "A minimum interference routing algorithm," 2004, pp. 1001–1005.

[10] I. Iliadis and D. Bauer, "A new class of online minimum-interference routing algorithms," in *Networking 2002, Proceedings of Second the International IFIP-TC6 Networking Conference*, May 19-24 2002, p. 959 ff.

[11] A. Tizghadam and A. Leon-Garcia, "AORTA: Autonomic network control and management system," 2008, pp. 1–4.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7266
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7266