Instituto de Engenharia de Sistemas e Computadores de Coimbra
Institute of Systems Engineering and Computers INESC - Coimbra

Jaime Silva, Teresa Gomes, Lúcia Martins,
and David Tipper

A note on link costs for selecting a spine considering path protection

No. 3 2015

ISSN: 1645-2631

Instituto de Engenharia de Sistemas e Computadores de Coimbra
INESC - Coimbra
Rua Antero de Quental, 199; 3000-033 Coimbra; Portugal
www.inescc.pt

The work of Jaime Silva was supported by ICIS Project CENTRO-07-0224-FEDER-0020003. The work Teresa Gomes and Lúcia Martins was in part financially supported by Fundação para a Ciência e a Tecnologia (FCT) under project grant UID/MULTI/00308/2013 and by ICIS Project CENTRO-07-0224-FEDER-0020003.

This page was intentionally left blank.

A note on link costs for selecting a spine considering path protection

Jaime Silva(1), Teresa Gomes(2,3), Lúcia Martins(2,3) and David Tipper(4)

(1)Centro de Informática da Universidade de Coimbra,
(2)Departamento de Engenharia Electrotécnica e de Computadores da FCTUC,
Pólo 2 da Univ. Coimbra, 3030-290 Coimbra, Portugal

(3)INESC-Coimbra, Rua Antero de Quental 199,
3000-033 Coimbra, Portugal.

(4)Graduate Telecommunications and Networking Program
University of Pittsburgh, Pittsburgh, PA, USA

Corresponding author: Teresa Gomes, teresa@deec.uc.pt

June 2015

Abstract

Communication networks resilience is very important considering the impact that disruption of communication services can have in today's society. In previous works the *spine* concept was introduced as an approach to ensure high availability to critical services while ensuring the possibility of very different levels of resiliency for less demanding services.

A spine is a subgraph of a network and in this work it will considered to be a spanning tree. The spine concept is discussed and approaches for a quick selection of a spanning tree (the spine), are presented. It is assumed end-to-end protection for all node pairs in the network, considering edge-disjoint path pairs, where the active path is routed on the spine.

The first the objective of this work is to evaluate if using a k-minimum spanning tree iterative algorithm is a more effective approach (considering the same link cost) than a previously proposed heuristic that requires only Kruskal's (or Prim's) algorithm. The second objective is to compare the performance of different link costs, seeking to identify the most effective cost, namely regarding average end-to-end availability, for obtaining an adequate admissible spine, considering a certain number k of sequentially generated spanning trees.

Regarding the first objective, using a relatively small number of k minimum spanning trees and a lexicographic version of Dijkstra's algorithm for calculating the backup path, allowed to improve previous results. Regarding the second objective, it was concluded that the k-betweenness centrality seems to the the metric with leads to better results.

1 Introduction

Nowadays communication networks are ubiquitous in our daily life, being one of the critical infrastructures that our society depends on. This leads to concerns about the reliability and resilience of communications when subjected to failures and attacks [19]. A communication network failure is defined as an event where it is not possible to deliver communication services by the network [19]. A network failure can typically occur due to cable cuts, natural disasters and physical/electronic attacks. Due to the importance of communication networks in todays society there is an interest in the design of resilient networks capable of surviving to failures. Usually network resilience can be improved when designing the network by placing sufficient diversity and redundancy in the network topology [19]. One important aspect in improving the network resilience, i.e. improving the network fault tolerance, is that not all the users need very high levels of availability. Indeed, the users that need a high level of availability are responsible for a small fraction of the total network traffic [10]. In fact, there is a need to support quality of resilience (QoR) classes [10] similar to the quality of services (QoS) classes. The QoR concept categorizes traffic in different levels of availability and fault protection for each class.

Earlier networks were engineered to offer only one type of service and hence only one level of resilience was provided. Later, the existence of different services led to different QoS classes with specific requirements for each class in multiservice networks. Similarly the QoR classes enable the distribution of the traffic trough links and nodes with different availability without the need of increasing the overall network reliability [5]. QoR has been a recent topic of research in the last years.

As the services evolve and become all-over-IP, QoR becomes an important topic of research. Hence, maintaining a network to meet a high level of resilience over the entire range of services, has very large of operational costs [5]. A solution to this problem is providing QoR classes by using different restoration mechanisms per traffic type. Moreover, by deploying different restoration mechanism at various network layers, QoR classes with different fault recovery capabilities and availabilities can be deployed [10]. Another approach can be seen in [18] where the authors present a typology of recovery methods and propose a framework of a multi-layer QoR. Recovering from a failure in the working path implies a sequence of operations were the original traffic is diverted to a fault-free recovery route. In [6] it is presented a classification of recovery methods based on several criteria. Also in [6] it is proposed a framework which enables the determination of QoR measures for the logical layer.

One important point in designing communications networks is that the network must be survivable, i.e. to be capable of continuous operation in the event of failures of nodes or links [20]. Network survivability can be enhanced by adding additional paths (backup paths) that are used in case of communication failure of the working path. The backup paths should be disjoint with the working path, in the nodes or in the edges. Depending on the approach used the backup paths can be pre-computed, i.e. protection is used, or calculated dynamically after the failure, i.e. restoration is used, as can be seen in [15] and the references therein.

In previous works a concept for a high availability portion of the network, termed *spine*, was presented [19, 10, 1]. The spine connects the nodes and links with traffic that needs a high level of availability and it is deployed at the physical layer. The higher availability of the spine, can be accomplished by using more expensive equipment and deploying locally redundant equipment in parallel. It was shown that the spine provides levels of availability differentiation at the physical level.It was also demonstrated that if carefully deployed the spine approach can provide higher overall average end-to-end availability for a given average edge availability [10]. The spine can be defined as a spanning tree connecting the edges where high availability communications services are needed. In [10] using the edge costs, based on a weighted linear combination of the edge betweenness centrality and the edge degree, and a heuristic based on the Kruskal's algorithm [13], a spine was generated. The concept of centrality stems from the social intuition that node/person is more central in the network if its is close to every other node/person in the network having more influence as can bee seen in [7] and in the references therein. The betweenness centrality comes also from social intuition and it is related with the nodes/persons that are more central if they stand between other nodes/person and are important for communication paths [7]. So the betweenness centrality is a measure of the centrality of a node in network and can be consider as the influence that a node has on the flow of communication trough the network. The betweenness centrality can be calculated using a diversity of methods, for instance using random walks [17] or using variants of the shortest paths [4]. It should be noted, that the betweenness centrality is one of several methods that measures the concept of centrality as can be seen in [3]. Hence, the edge betweenness centrality, which is akin to the betweenness centrality, measures the influence of an edge on the communication between pairs of vertices in a network. The edge betweenness centrality is then used to generate the spine in conjugation with the edge degree which is a simple measure for the influence of each edge in the network, i.e., in a fail state edges with high degree will cause disruption a larger number of communication paths than edges with smaller degree.

It is shown that, using the same edge costs proposed in [10] and also using two new costs, one based on the principal eigenvector of the adjacency matrix [16, 2], related to the edge centrality and the other based on the k-betweenness centrality [11], that an admissible spine can be generated by using a relatively small number of k minimum spanning trees. More precisely, to generate candidate spines the algorithm proposed by Katoh et al [12, 8] for the iterative generation of spanning trees by increasing cost, was used. A spine is only considered admissible if for every end-do-end path on the spine an edge-disjoint backup path can be obtained.

The first objective of this work was to evaluate if using a k-minimum spanning tree iterative algorithm is a more effective approach (considering the same link costs) than a previously proposed heuristic [10] which requires only Kruskal's (or Prim's) algorithm. The spine is evaluated, among other performance measures, regarding the average end-to-end availability (considering path protection), its diameter, and the average number of hops of the WPs. The second objective is to compare the performance of different link costs, seeking to identify the most effective cost for obtaining an adequate admissible spine, considering a limited number k of sequentially generated spanning trees,

Regarding the first objective, using a relatively small number of k minimum spanning trees and a lexicographic version of Dijkstra's algorithm for calculating the backup path, it was possible to improve previous results. This approach allows the user to have greater control over the number of candidate spines, than the approach proposed in [10]. Regarding the second objective, it was concluded that the k-betweenness centrality seems to the metric with leads to better results.

It is also studied the effect of having uniform availabilities, i.e., predefined availabilities for edges on the spine and off the spine, in the considered spine performance measures. Finally, the spine performance measures are evaluated for the case of non uniform availabilities, i.e., variable edges availabilities.

This work is organized as follows. In section 2 notation is introduced, including metrics used to characterize the spine are presented. In section 3 the cost introduced in [10] are presented, together with two other considered link costs. The results and discussion of several spine performance metrics are described in section 4 using uniform and non uniform availabilities. Finally the conclusions are outlined in the last section.

2 Notation

The notation used in this report is listed here.

Sets

$\mathcal{N}$ - set of physical nodes in the graph.

$\mathcal{L}$ - set of physical links in the graph (undirected edges).

$\mathcal{G}$ - network graph: $\mathcal{G} = (\mathcal{N}, \mathcal{L})$.

$\mathcal{F}$ - set of end-to-end flows

$\mathcal{S}$ - set of links in the spine.

$\mathcal{G}_\mathcal{S}$ - network subgraph defined by the spine, $\mathcal{G}_\mathcal{S} = (\mathcal{N}, \mathcal{S})$.

Indexes

n - node index.

l - link (edge) index ($l \in \mathcal{L}$) connecting i and j with $i, j \in \mathcal{N}$.

f - a bidirectional symmetric flow ($f \in \mathcal{F}$).

s, d - source and destination nodes for each flow ($s, d \in \mathcal{N}$).

Paths

WP_f - Working Path for flow f.

BP_f - Backup Path for flow f.

Availability

a_l - availability of link l.

A_f^{WP} - Working Path availability for flow f:

$$A_f^{WP} = \prod_{l \in WP} a_l \tag{1}$$

If a shortest path algorithm is used to obtain the most available path, the cost of each link l must be defined as $-\log(a_l)$, transforming the problem of maximizing the availability into a minimization problem of additive costs.

A_f^{BP} - Backup Path Availability for flow f (similar to equation (1)).

A_f - availability of flow f. Assuming WP_f and BP_f are edge-disjoint, $A_f = 1 - (1 - A_f^{WP})(1 - A_f^{BP})$.

Structural measures

The next two topological/structural measures (eb_l, $BC_k(l)$) can be weighted or unweighted. This will clarified in the text.

eb_l - edge l betweenness centrality which is determined from:

$$eb_l = \frac{2}{|\mathcal{N}|(|\mathcal{N}| - 1)} \sum_{s,d \in \mathcal{N}, s \neq d} \frac{\sigma(s, d|l)}{\sigma(s, d)} \tag{2}$$

where $\sigma(s,d)$ is the number of shortest paths between nodes s and d and $\sigma(s,d|l)$ is the number of those paths that use edge l. To compare the edge betweenness centrality for different networks a normalization factor, $\frac{2}{|\mathcal{N}|(|\mathcal{N}|-1)}$, was applied.

$BC_k(l)$ - k-betweenness centrality, adapted from Jiang et. al [11],

$$BC_k(l) = \sum_{s,d \in \mathcal{N}, s \neq d} \frac{\sigma_k(s,d|l)}{\sigma_k(s,d)} \tag{3}$$

where $\sigma_k(s,d|l)$ is the number of shortest paths between nodes s and d, that uses edge l, whose length is less than or equal to the length of the shortest path, between nodes s and d, plus k. Likewise, $\sigma_k(s,d)$ is the number of shortest paths between nodes s and d whose length is less than or equal to the length of the shortest path plus k.

ed_l - degree of edge l, defined as the sum of the degree of the edge's end nodes.

c_l - cost of using edge l.

Performance measures

$A_{\mathcal{S}}$ - average value of A_f when the WP is on the spine.

$A_{\mathcal{S}}^{WP}$ - average value of A_f^{WP} when WP is on the spine.

$A_{\mathcal{S}}^{BP}$ - average value of A_f^{BP} when WP is on the spine.

$eb_{\mathcal{S}}$ ($eb_{\mathcal{G}}$) - average value of eb_l in $\mathcal{G}_S$ ($\mathcal{G}$), considering the edges in $\mathcal{S}$ ($\mathcal{L}$).

$h_{\mathcal{S}}$ ($h_{\mathcal{G}}$) - average shortest paths in $\mathcal{G}_{\mathcal{S}}(\mathcal{G})$, i.e. the sum of the number of hops between (s,d) in $\mathcal{G}_{\mathcal{S}}(\mathcal{G})$, divided by the number of shortest paths between (s,d) in $\mathcal{G}_{\mathcal{S}}(\mathcal{G})$.

$ed_{\mathcal{S}}$ ($ed_{\mathcal{G}}$) - average of ed_l over all edges in $\mathcal{G}_{\mathcal{S}}(\mathcal{G})$.

$di_{\mathcal{S}}$ ($di_{\mathcal{G}}$) - spine (Graph) diameter, that is the length (hops) of the longest shortest path in $\mathcal{G}_{\mathcal{S}}(\mathcal{G})$.

3 Generating candidate spines

To obtain different spines, first four sets of the costs of the links were defined as a weighted linear combination related to the edge betweenness centrality and the edge degree, similarly to what was done in [10]. Two additional costs were also considered, which are only related with the edge centrality.

The costs of the edges c_l^i, with $i \in \{A, B, C, D, E, F\}$ were defined as follows, with $O \leq \alpha \leq 1$:

- Case A: for a given $\alpha > 0$, the larger the edge degree and the larger the edge betweenness centrality, the smaller the cost of the edge l:

$$c_l^A = (1-\alpha)\frac{(\min_l\ ed_l)}{ed_l} + \alpha\frac{(\min_l\ eb_l)}{eb_l} \tag{4}$$

- Case B: for a given $\alpha > 0$, the larger the edge degree and the smaller the edge betweenness centrality, the smaller the cost of edge l:

$$c_l^B = (1-\alpha)\frac{(\min_l\ ed_l)}{ed_l} + \alpha\frac{eb_l}{(\max_l\ eb_l)} \tag{5}$$

- Case C: for a given $\alpha > 0$, the smaller the edge degree and the larger the edge betweenness centrality, the smaller the cost of edge l:

$$c_l^C = (1-\alpha)\frac{ed_l}{(\max_l\ ed_l)} + \alpha\frac{(\min_l\ eb_l)}{eb_l} \tag{6}$$

- Case D: for a given $\alpha > 0$, the smaller the edge degree and the smaller the edge betweenness centrality, the smaller the cost of edge l:

$$c_l^D = (1-\alpha)\frac{ed_l}{(\max_l\ ed_l)} + \alpha\frac{eb_l}{(\max_l\ eb_l)} \tag{7}$$

- Case E: the larger the edge centrality, calculated using the principal eigenvector v

of the adjacency matrix, the smaller the edge cost of the edge l;

$$c_l^E = \frac{2}{v_i + v_j} \tag{8}$$

where $i, j \in \mathcal{N}$ and v_i and v_j corresponds to the i, j components of the principal eigenvector, respectively, and to the edge l head and tail nodes respectively. The eigenvector centrality v_i of a vertex i in an unweighted network is defined to be proportional to the sum of the centralities of the vertex's neighbors. Hence, a vertex can acquire high centrality by connecting to a high number of vertices or by connecting to others that are highly central [16, 2].

- Case F: the larger the edge centrality, calculated using the k-betweenness centrality $BC_k(l)$ with $k = 1$, the smaller the edge cost of the edge l:

$$c_l^F = \frac{\min_l\ BC_k(l)}{BC_k(l)} \tag{9}$$

Two type of scenarios will be considered: uniform availabilities, i.e., predefined availabilities for edges on the spine and off the spine, and non-uniform availabilities where the value of an edge availability depends on the distance between the edge's end nodes.

In the case of uniform availabilities eb_l and $BC_k(l)$ are calculated considered an *unweighted graph*, while in the case of non-uniform availabilities eb_l and $BC_k(l)$ are calculated considered a *weighted graph*.

The algorithm proposed by Katoh et al [12, 8] for the iterative generation of spanning trees by increasing cost was applied. A candidate spine (spanning tree) is only considered *admissible* if an edge-disjoint path can be found in the network for each active path in the spine. However, while in [1] the BP was calculated as the min-hop (fully disjoint path with the WP), *avoiding* the edges of the spine if possible, in the present work it was considered the BPs *can use* edges on the spine:

1. In the case of uniform availabilities (sub-section 4.2), the backup path was calculated on the graph where the edges of the working path routed on the spine have being

removed, using a lexicographic version of the Dijkstra algorithm, with the first edge cost equal to one and the second cost being the edge availability which takes the value of 0.99 for edges off the spine and 0.999 for edges on the spine.

Therefore in the case of a tie the backup path with edges on the spine will be favored, which may result in larger availability than obtained in [1] (where the edges in spine were avoided by the backup paths) or in [9] (where every BP is simply a min-hop path in the corresponding sub-graph).

2. Similarly, in the case of non-uniform availabilities, the backup path was calculated on the graph disjoint with the edges of the working path in the spine using a lexicographic version of the Dijkstra algorithm, where the first objective is obtaining a min-hop path, and the second objective is maximizing the path availability.

Additionally, in sub-section 4.4 results are presented considering (as in [1]) that the BP *must avoid* the edges in the spine, as much as possible. However, while avoiding the edges in the spine, we still use a lexicographic version of the Dijkstra's algorithm to obtain, among BPs with the same number of hops, the one with largest availability.

4 Results and Discussion

4.1 Networks topology measures

Some topological characteristics of the networks used in this study are presented in Table 1.

As can be seen there are networks with a small average degree number, $< k >$, for instance nsf. There also networks with an higher average degree like newyork, that shares the same number of nodes with epan16. In this study are also presented results for larger networks like italia and germany50.

Table 1: Network topological information

Network	**Nodes**	**Edges**	**Number of spanning trees**	$<\mathbf{k}>$	eb_G	ed_G	h_G	di_G
polska	12	18	5161	3.0	0.1187	6.3333	2.1364	4
nsf	14	19	5862	2.7143	0.1180	5.7895	2.2418	4
epan16	16	23	43.7E03	2.8750	0.1149	6.0870	2.6417	6
newyork	16	49	1.4538E10	6.1250	0.0350	13.5510	1.7167	3
italia	32	69	53.3E14	4.3125	0.0425	9.4493	2.9315	6
germany50	50	88	4.5872E19	3.520	0.0460	7.65909	4.0482	9

4.2 Results using the c_l^A to c_l^D costs

In this sub-section are evaluated and compared several spine performance measures, for spines generated using edge costs c_l^A to c_l^D.

The number of trees to consider is this study is a relevant parameter. It was decided to generate $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees for each network. It is recalled that for this purpose the algorithm proposed by Katoh et al [12, 8] for the iterative generation of spanning trees by increasing cost was used.

Note that considering the number of k trees equal to $|\mathcal{N}|\,|\mathcal{L}|$ results in considering a very different proportion of the total number of existing trees when $|\mathcal{N}|$ or $|\mathcal{L}|$ grow.

4.2.1 Ratio of discarded trees

One of the objectives of this work is identifying an edge cost that will allow to obtain admissible candidate spines with good performance. If the number of non-admissible spines (discarded trees) is too high then one may have very few candidates to select from. Hence in this section the ratio between the number of discarded spanning trees, each one corresponding to a potential spine and the total number of generated spanning trees ($|\mathcal{N}|\,|\mathcal{L}|$) for each network, using a k-minimum spanning tree algorithm, is presented.

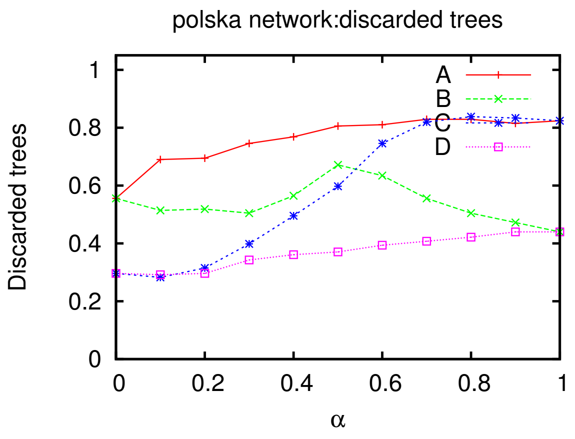
As already mentioned, the working path is on the spine. The graph where the backup path will be calculated corresponds to the original graph where the cost of the edges of the WP are changed to a sufficiently large cost. The backup path (disjoint with the edges of the working path) is calculated using a lexicographic version of the Dijkstra algorithm, with the first cost calculated using a edge cost equal to one and the second cost being the edge availability which takes the value of 0.99 for edges off the spine and 0.999 for edges on the spine.

For sparser networks like Figure 1a, 1b it can be seen that the number of discarded spanning trees is high, for Case A and C, when α is larger than 0.5, i. e., when high betweenness edges are most likely to be part of the spanning tree. In contrast, in Figure 1c it is also possible to observe that the number of discarded spanning trees diminishes with the increase of α for Case A. Nevertheless, for $\alpha > 0.5$, the other three cases (B, C, D) present, in general, similar behavior for polska, epan16 and nfs networks:

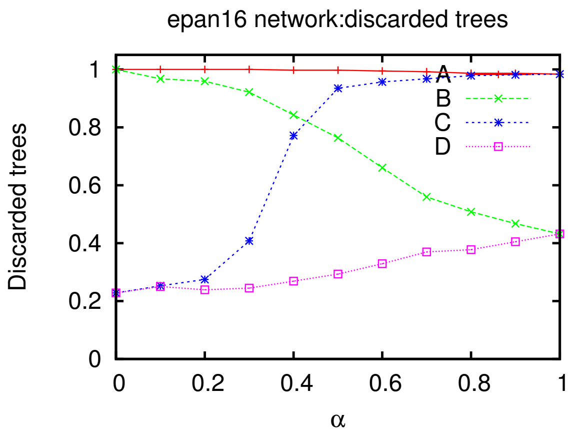
- decrease in the ratio of discarded trees with α for Case B, starting (for $\alpha = 0$) with a large value of discarded trees when edges of high degree are preferentially included in the spine;
- increase in the ratio of discarded trees with α for case C, specially for $\alpha \geq 0.5$;
- slight increase in the ratio of discarded trees with α for Case D.

For Figure 2a, which is a denser network, none of the generated $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees generated was discarded. Also for a denser network like italia, from Figure 2b it can be observed that only for high values of α there is a high number of discarded spanning trees, when using Cases A and C. However the results for germany50, shown in Figure 2c, are quite different for intermediate values of α, for all cases except D, where the spanning trees will mostly contain low degree edges and/or low edge betweenness edges, and the almost all spanning trees are admissible, regardless of the value of α . For $\alpha = 1$ the results for germany50 and italia are similar for all Cases. In germany50, it can be seen from that using the cost c_l^C discards a very high number of spanning trees with increasing α, in opposition to the case B which decreases the number of discarded spanning trees with the increase of α. This is in agreement with the fact that selecting edges with low degree edges and/or low edge betweenness leads to admissible spanning trees.

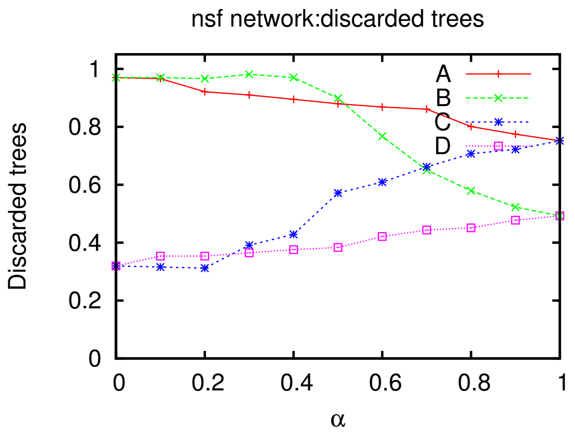
It is possible to say that, in this study, cost c_l^D prevails over the others, discarding a smaller number of spanning trees, because the first generated spanning trees will contain



(a) polska



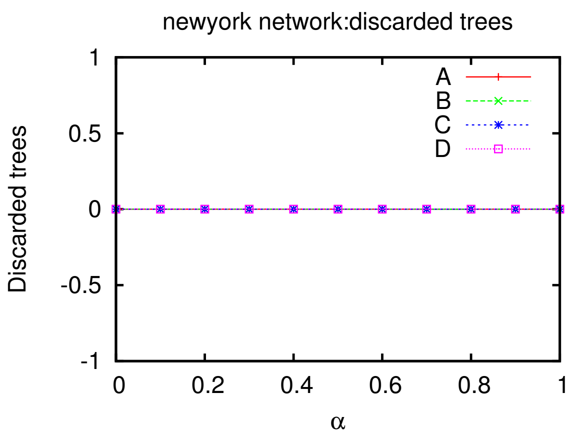
(b) epan16



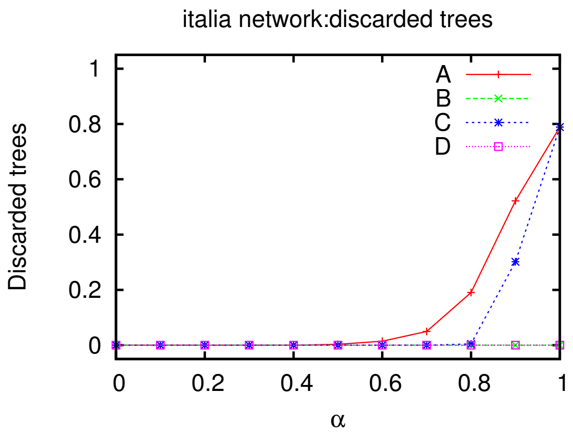
(c) nsf

Figure 1: Ratio of discarded spanning trees for different α values using the generated $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees on sparser networks.

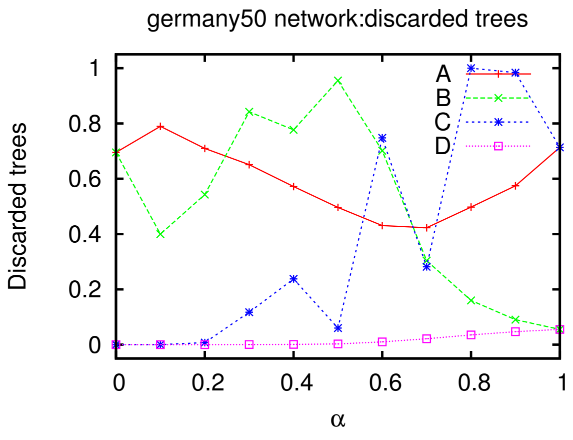
many edges with low edge betweenness and low degree. In the opposite extreme is cost c_l^A which will result in spanning trees with edges with high degree and high edge betweenness,



(a) newyork



(b) italia



(c) germany50

Figure 2: Ratio of discarded spanning trees for different α values using the generated $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees on denser networks.

often including all edges adjacent to a high centrality node, making it impossible to find an edge-disjoint path for one (or more) of the WPs in the selected spine.

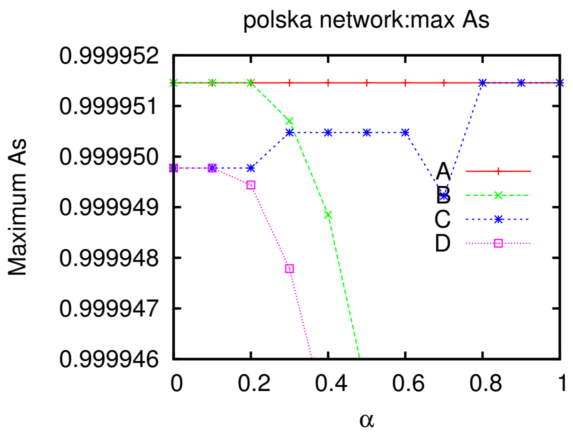
4.2.2 Maximum of $A_{\mathcal{S}}$ using the spine

The most relevant measure of the performance of an admissible spine is its $A_{\mathcal{S}}$. As in the previous subsection, $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees were generated, using the different edge weights, see Equations (4)-(7). For each generated spanning tree, with an existing edge-disjoint backup path for each WP in the tree, $A_{\mathcal{S}}$ was calculated, and the maximum obtained value of $A_{\mathcal{S}}$ was registered (from the $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees). Recall the edge availability was 0.99 for the edges off the spine and 0.999 for the edges on the spine. The working path was considered to be on the spine. A disjoint backup path was calculated a lexicographic Dijkstra with a edge cost equal to one for the first cost and equal to the edge availability for the second cost – *hence not avoiding the edges in the candidate spine.*

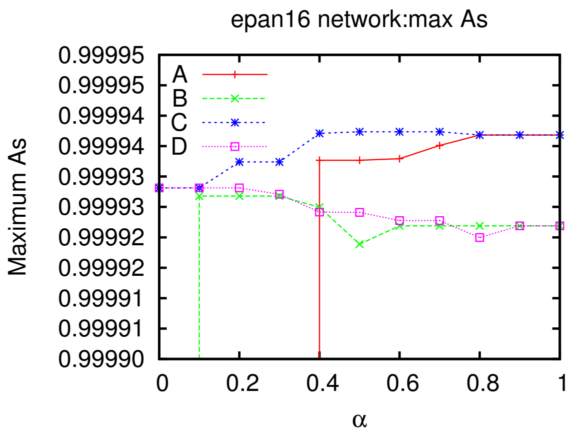
Observing the value of the maximum of $A_{\mathcal{S}}$ for different α values in the sparser networks, presented in Figure 3a, 3b and 3c, it can be concluded that using the costs c_l^A and c_l^C for the edge weights, the maximum of $A_{\mathcal{S}}$ is higher when comparing with the other edge weight metrics.

Also in Figure 3a and 3b it can be observed that the higher values of the maximum of $A_{\mathcal{S}}$ are obtained for high values of α (in cases A and C) indicating the prevalence of the edge betweenness centrality as the topological metric with best results. Nevertheless, it can seen that the maximum of $A_{\mathcal{S}}$ observed in Figures 3c and 3b are obtained using the cost c_l^C with values of α around 0.5. In fact, its possible to observe in Figures 3c and 3b that is the combination of a low edge degree with large edge betweenness that produces the best results for the maximum of $A_{\mathcal{S}}$.

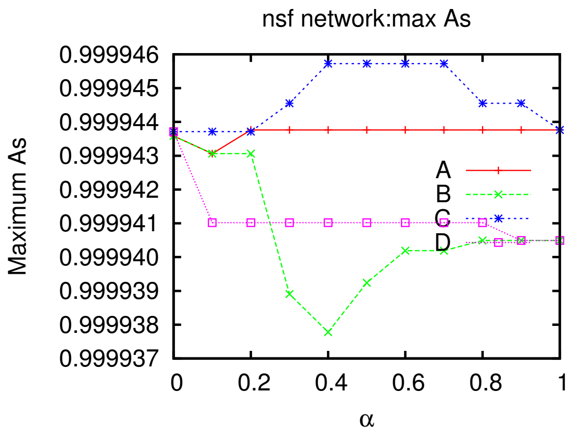
For the denser networks presented in Fig. 4a, 4b and 4c it is possible to observe that the edges weight generated with metric c_l^A consistently presents the higher values for the maximum of $A_{\mathcal{S}}$. The only exception to the latter statement is the slight decrease of case A, observed in Figure 4b, compared with C. This decrease can be attributed to an high number of discarded spanning trees for the cost c_l^A comparing with the cost c_l^C as can be observed in Figure 2b. Nevertheless, its important to stress that effect of the edge degree is minimum, for the denser networks, being the edge betweenness the most important



(a) polska



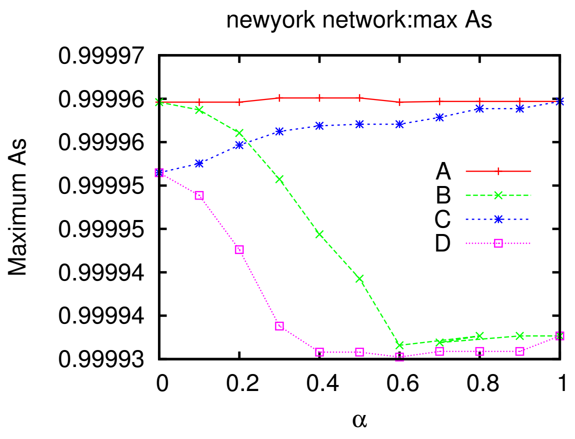
(b) epan16



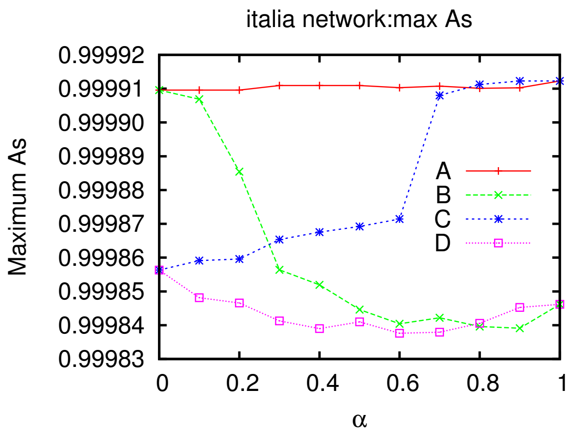
(c) nsf

Figure 3: Maximum of $A_{\mathcal{S}}$ for different α values using the generated $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees on sparser networks.

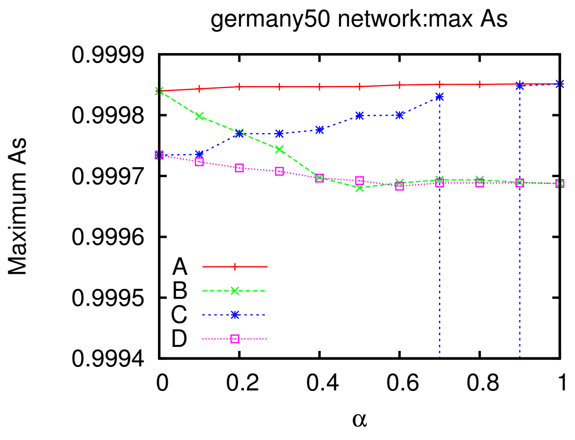
metric when defining the cost. In fact, for the denser networks using the cost c_l^A, it can be seen that the edges with low edge betweenness will be selected in later iterations of



(a) newyork



(b) italia



(c) germany50

Figure 4: Maximum of $A_{\mathcal{S}}$ for different α values using the generated $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees on denser networks.

the k-minimum spanning tree generation algorithm.

By selecting the edges with a high value of edge betweenness first, the WPs tend to be shorter, thus increasing A_S. On the other hand as the BP does not have to avoid the edges in the spine, the BP also tends to be shorter and more reliable. Nevertheless, due to the fact that the backup path must be disjoint with the active path, the presence of these edges with high edge betweenness in the spine indicates that in the calculation of the backup it can be impossible to define a disjoint backup path, increasing the number of discarded trees.

4.2.3 Maximum of the minimum of A_f

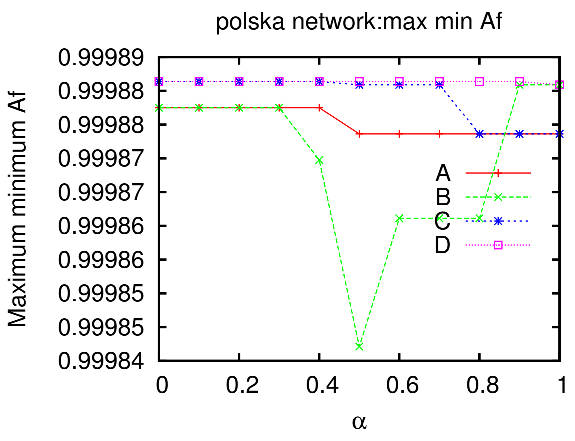
Considering path protection, another relevant measure of the performance of the spine is the maximum of minimum of A_f were a small value indicates the worst case[1] in the calculation of A_f. The results for the maximum of the minimum of A_f for the sparser networks are presented in Figures 5a, 5b and 5c. It can be observed from these figures that the cases D and C have the larger values for the maximum of the minimum of A_f. Nevertheless, the cost c_l^C is slightly higher in Figure 5c for $\alpha \in [0.4, 0.7]$.

For denser networks the value of the maximum of the minimum of A_f is higher for the cases A and C, as can be seen in Figures 6a, 6b and 6c, while case D presents the poorest performance. In the same Figures is possible to observe that the higher values for the maximum of the minimum of A_f is achieved when the α is high. This indicates that edge betweenness centrality is preponderant for achieving high values of maximum of the minimum of A_f. Comparing the maximum of the minimum of A_f presented in Figure 6a, 6b and 6c with the maximum of A_S discussed in the previous subsection it can be stated that the cost c_l^A is the weight which results in spines with the best results in the case of denser networks.

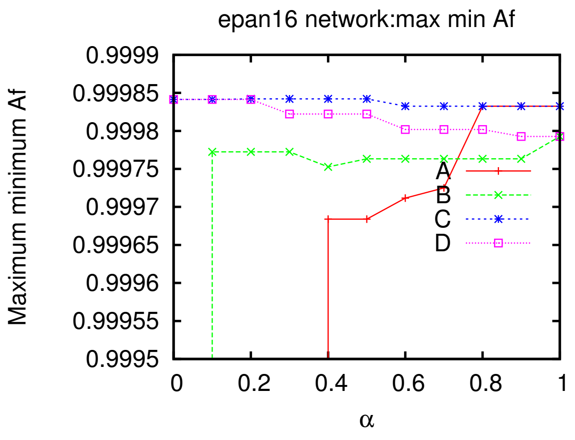
4.3 Two additional cost: using the c_l^E to c_l^F costs and the c_l^A cost

In this section results concerning the generation of $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using the metrics c_l^E, c_l^F and c_l^A (with $\alpha = 1.0$) as edges weights, are presented. The value of $\alpha = 1.0$ was chosen due to the fact that it corresponded to the spines with larger A_S,

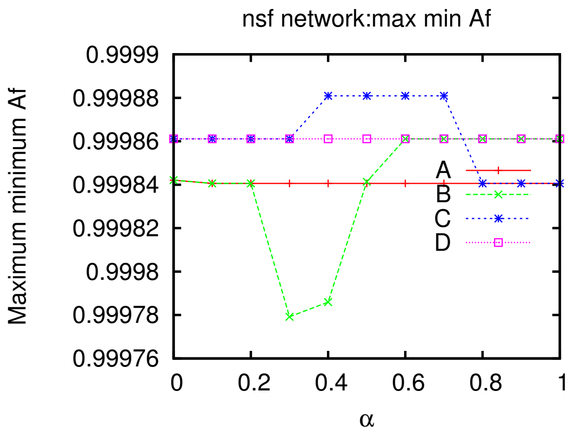
[1]In fact a fairness index could have been calculated instead.



(a) polska



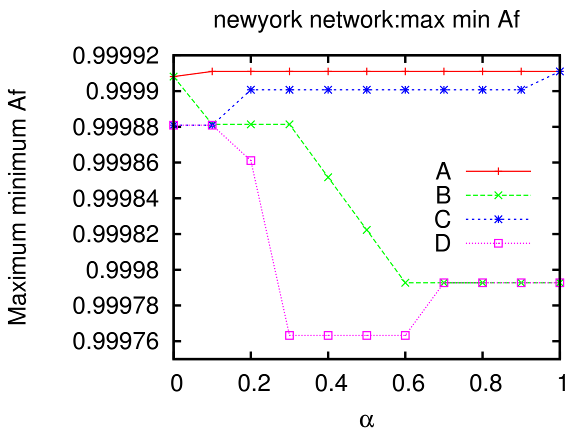
(b) epan16



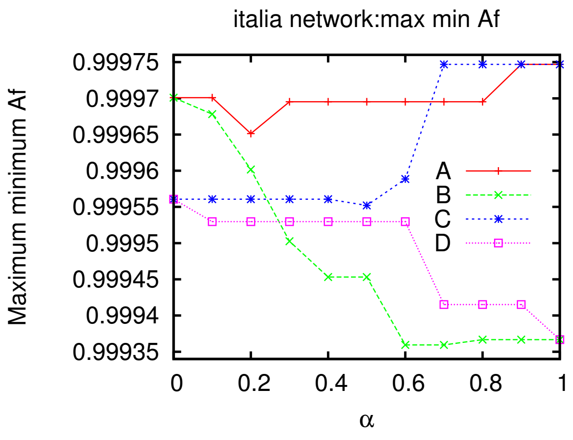
(c) nsf

Figure 5: Maximum of the minimum of A_f for different α values using $|\mathcal{N}|\,|\mathcal{L}|$ generated minimum spanning trees on sparser networks.

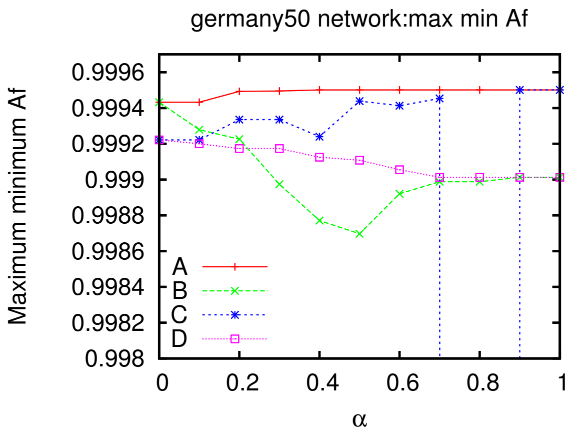
max of min A_f in four of the six tested networks, albeit the large number of discarded spanning trees.



(a) newyork



(b) italia



(c) germany50

Figure 6: Maximum of the minimum of A_f for different α values using $|\mathcal{N}|\,|\mathcal{L}|$ generated minimum spanning trees on denser networks.

In the results presented in Table 2 for the cost c_l^A, Table 3 for the cost c_l^E and in Table 4 for the cost c_l^F, each row represents a spanning tree with a maximum value (marked as bold) for the designated performance metric ($A_{\mathcal{S}}^{WP}$, $A_{\mathcal{S}}$ and $\max\ \min A_f$). When a metric exhibits a maximum value in several spanning trees the spanning tree with maximum $A_{\mathcal{S}}$ is selected.

Table 2: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using as cost c_l^A with $\alpha = 1.0$.

Network	**Discarded spanning trees**	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	$\max\ \min A_f$
polska	0.8241	0.9972757	**0.9999515**	0.2479	5.0909	2.7273	5	0.99985179
		0.9973210741	0.9999485	0.2438	5.4545	2.6818	5	0.9998034
		0.9972003	0.9999503	0.2548	5.0909	2.8030	6	**0.9998736**
nsf	0.7519	**0.9967411**	**0.9999438**	0.2511	4.9231	3.2637	6	0.9998034
		0.9966426	0.9999431	0.2587	4.9231	3.3626	7	**0.9998406**
epan16	0.9836	**0.9965554**	**0.9999368**	0.23	4.6667	3.45	7	**0.9998325**
newyork	0.0	0.9973610	**0.9999597**	0.1761	7.0667	2.6417	5	0.9998814
		0.9975771	0.9999587	0.1617	9.0667	2.425	5	0.9998814
		0.9974441	0.9999588	0.1706	8.1333	2.5583	5	**0.9999110**
italia	0.7889	0.9959491	**0.9999123**	0.1309	5.8065	4.05847	8	**0.9997469**
		0.9960515	0.9999073	0.1276	6.2581	3.9556	8	0.9996441
germany50	0.7136	0.9945576	**0.9998514**	0.1114	5.5102	5.4571	12	0.9994649
		0.9947693	0.9998507	0.1070	5.6327	5.2441	11	0.9994333
		0.9945574	0.9998505	0.1114	5.4286	5.4571	11	**0.9995012**

It can be observed, in Tables 2, 3 and 4, that the fraction of discarded spanning trees is smaller for the cost c_l^E. Nonetheless, for the epan16 network using the c_l^E discards all the generated spanning trees indicating that the total number of trees, $|\mathcal{N}|\,|\mathcal{L}|$, is too small. It can also be observed, by comparing the results in Tables 2, 3 and 4, that using cost defined as c_l^A, with $\alpha = 1.0$, for generating the k-minimum spanning trees results in the higher value for the $A_{\mathcal{S}}$ performance metric, with the exception of the results for the network epan16 where the c_l^F is better for the $A_{\mathcal{S}}$ performance metric, albeit with a larger value for the $h_{\mathcal{S}}$ and $di_{\mathcal{S}}$. For the $A_{\mathcal{S}}^{WP}$ using the cost defined as c_l^E produces better results for all the tested networks with the exception of the results for the epan16 network where the c_l^A and c_l^F have better results and for germany50 where c_l^A achieves the largest value for $A_{\mathcal{S}}^{WP}$.

Comparing the $\max\ \min A_f$ performance metric, i.e, the maximum of the minimum of the A_f for each spine using the c_l^A, c_l^E and c_l^F for generating the $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees, it is possible to conclude that for sparser networks the metric c_l^E produces the best results, albeit, discarding all the trees for the epan16 network. For denser networks metric

Table 3: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using as cost c_l^E

Network	**Discarded spanning trees**	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	$\max\ \min A_f$
polska	0.5648	0.9972757	**0.9999515**	0.2479	5.0909	2.7273	5	0.9998518
		0.9973362	0.9999479	0.2424	5.2727	2.6667	5	0.9998373
		0.9971398	0.9999479	0.2603	4.9091	2.8636	6	**0.9998775**
nsf	0.9323	0.9967192	**0.9999436**	0.2527	4.7692	3.2857	7	**0.9998421**
		0.9968286	0.9999435	0.2443	4.6154	3.1758	6	0.9997890
epan16	1.0	-	-	-	-	-	-	-
newyork	0.0	0.9976352	**0.9999595**	0.1578	9.0667	9.0667	4	0.9998814
		0.9977267	0.9999579	0.1517	10.1333	2.275	4	0.9998814
		0.9975355	0.9999579	0.1644	8.4	2.4667	5	**0.9999007**
italia	0.0	0.9960596	**0.9999094**	0.1273	6.3871	3.9476	9	0.9996441
		0.9962724	0.9999035	0.1204	6.9677	3.7339	8	0.9996441
		0.9960296	0.9999022	0.1283	6.7742	3.9778	9	**0.9996513**
germany50	0.0035	0.9945680	**0.9998379**	0.1112	5.2245	5.4465	11	0.9993800
		0.9946345	0.9998321	0.1098	5.3061	5.3796	10	0.9992682
		0.9943814	0.9998323	0.115	5.2245	5.6343	11	**0.9994220**

c_l^A produced the best results for the $\max\ \min A_f$ performance metric.

Table 4: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using as cost c_l^F

Network	**Discarded spanning trees**	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	$\max\ \min A_f$
polska	0.7407	**0.9972757**	**0.9999515**	0.2479	5.0909	2.7273	5	0.9998518
		0.9970946	0.9999498	0.2645	5.0909	2.9091	6	**0.9998692**
nsf	0.8421	**0.9967411**	**0.9999438**	0.2511	4.9231	3.2637	6	0.9998034
		0.9966426	0.9999431	0.2587	4.9231	3.3626	7	**0.9998406**
epan16	0.9239	0.9964808	**0.9999373**	0.235	4.6666	3.525	8	0.9997724
		0.9965554	0.9999368	0.23	4.6667	3.45	7	**0.9998325**
newyork	0.0	0.9976186244	**0.9999596**	0.1589	8.9333	2.3833	4	0.9998814
		0.9977267	0.9999586	0.1517	10.1333	2.275	4	0.9998814
		0.9974026	0.9999575	0.1733	7.7333	2.6	5	**0.9999007**
italia	0.7319	0.9961981	**0.9999104**	0.1229	6.7742	3.8085	8	0.9996953
		0.9962281	0.9999065	0.1219	6.7742	3.7782	7	0.9996953
		0.9960896	0.9999097	0.1264	6.5806	3.9173	8	**0.9997469**
germany50	0.68	0.9946767	**0.9998505**	0.1089	5.4694	5.3371	11	0.9994333
		0.9947189	0.9998461	0.1081	5.5918	5.2947	10	0.9994333
		0.9945301	0.9998428	0.1119	5.5918	5.4849	12	**0.9994954**

It can be concluded that in general using the costs based in the edge betweenness centrality or k-betweenness centrality produce the best values for the considered performance measures.

Table 5: Results considering all the spanning trees presented in [10], backup path can use the spine edges.

Network	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$
polska	0.99720	**0.9999495**	0.2548	5.0909	2.8030	6
	0.99734	0.9999451	0.2424	5.2727	2.6667	5
nsf	0.99637	**0.9999445**	0.2798	4.6154	3.6374	9
	0.99687	0.9999406	0.2409	4.6154	3.1319	6
epan16	0.99650	**0.9999332**	0.2339	4.6667	3.5083	8
	0.99670	0.9999234	0.2200	5.0667	3.3000	7

4.4 Comparison with previous work

By generating all the spanning trees it is possible to compare the results of the performance metrics with previous work as in [10]. In Table 5 it is presented a summary of the results discussed in [10] using all the spanning trees. The bold values represents the maximum of the corresponding metric. The first line in the table corresponds to the spanning tree with a maximum value for the first metric ($A_{\mathcal{S}}^{WP}$); the second line is related with the spanning tree that haves a maximum value in the second metric ($A_{\mathcal{S}}$) and the third line corresponds to the maximum of the last metric ($\max\ \min A_f$). It should be noted that when the spanning tree has a maximum value in the two metrics only, one line is presented. Only the networks with the smaller number of spanning trees – see Table 1 – were used. It is important to stress that in [10] the calculation of the backup path a standard implementation of the Dijkstra algorithm was used, while in the results presented in this work a lexicographic version of the Dijkstra's algorithm was considered. It is possible to observe in Table 6 that the maximum value for $A_{\mathcal{S}}$ is higher when comparing with the results presented in [10], reproduced in Table 5. Again in Table 6 the bold values indicates the maximum value for the metric. When a metric exhibits a maximum value in several spanning trees, the spanning tree with maximum $A_{\mathcal{S}}$ is selected.

It is also possible to observe that the maximum values of $A_{\mathcal{S}}^{WP}$ remain the same due to the fact that the working path depends only of the selected spine. The difference that is found by comparing the maximum value of $A_{\mathcal{S}}$ with the previous work can be explained by recalling that in this work for the calculation of the backup path a lexicographic version of the Dijkstra algorithm is used. The implemented lexicographic Dijkstra algorithm minimizes the hop count and then maximizes the availability. It is important to stress that in this work and in [10] the working path and the backup path are edge-disjoint.

A different approach (similar to the one considered in [1]) for the calculation of the backup is instead of considering all edges equally (including those in the spine that are not in the working path) one may try to avoid whenever possible the spine edges when routing the backup path. The latter approach can be deployed for classes with different QoR. The results using this approach are presented in Table 7. It is possible to observe that avoiding the spine edges in the calculation of the backup path decreases the maximum value of $A_{\mathcal{S}}$ for small networks like the polska network. For larger networks there is no difference for the maximum value of $A_{\mathcal{S}}$ when the calculation of the backup path can use the spine edges (Table 6) or avoids then (Table 7). Comparing in Table 7 the maximum of max min A_f with the same metric in Table 6 it can be observed that there is no difference between these values. In fact, with exception of the $A_{\mathcal{S}}$ for polska network there is no difference between the two cases. The difference in the maximum value of $A_{\mathcal{S}}$ for the polska network can be explained due to an increase in $h_{\mathcal{S}}$ and $di_{\mathcal{S}}$ when avoiding the spine as can be seen by comparing the values of Table 7 with Table 6.

Table 6: Results considering all the spanning trees, backup path can use the spine edges

Network	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f
polska	0.99728	**0.9999515**	0.2479	5.0909	2.7273	5	0.9998518
	0.99734	0.9999479	0.2424	5.2727	2.6667	5	0.9998373
	0.99646	0.9999438	0.3223	4.1818	3.5455	8	**0.9998819**
nsf	0.99637	**0.9999457**	0.2798	4.4615	3.6374	9	0.9998809
	0.99687	0.9999410	0.2409	4.6154	3.1319	6	0.9997642
	0.99637	0.9999457	0.2798	4.4615	3.6374	9	**0.9998809**
epan16	0.99648	**0.9999373**	0.2350	4.6667	3.525	8	0.9997724
	0.99670	0.9999251	0.2200	5.0667	3.3000	7	0.9997507
	0.99502	0.9999258	0.3328	4.1333	4.9917	13	**0.9998543**

Table 7: Results considering all the spanning trees. The backup path avoids the spine edges

Network	$A^{WP}_{\mathcal{S}}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max $\min A_f$
polska	0.99702	**0.9999511**	0.2713	5.0909	2.9848	7	0.9998692
	0.99734	0.9999468	0.2424	5.2727	2.6667	5	0.9998373
	0.99646	0.9999438	0.3223	4.1818	3.5455	8	**0.9998819**
nsf	0.99637	**0.9999457**	0.2798	4.4615	3.6374	9	0.9998809
	0.99687	0.9999410	0.2409	4.6154	3.1319	6	0.9997642
	0.99637	0.9999457	0.2798	4.4615	3.6374	9	**0.9998809**
epan16	0.99648	**0.9999373**	0.2350	4.6667	3.525	8	0.9997724
	0.99670	0.9999251	0.2200	5.0667	3.3000	7	0.9997507
	0.99502	0.9999258	0.3328	4.1333	4.9917	13	**0.9998543**

4.5 Results for the spine using non uniform availabilities

The results presented in this section uses the edge availability a_l calculated using the following Equation [14]:

$$a_l = 0.99987^{d_l/(250\times 1.6093)} \qquad (10)$$

In Equation (10) d_l is the distance between the two end nodes of edge l in the network (calculated assuming that the coordinates of the nodes correspond to their GPS locations). Also in this section a *weighted* version of eb_l (equation (2)) that considers the edge availability a_l in the calculation of the shortest paths, is used to determine the spine. The weighted eb_l values are used as edge weights, more precisely $1/eb_l$, in the algorithm proposed by Katoh et al [12, 8] for the iterative generation of spanning trees by increasing cost, designated here as $c_l^{A'}$.

In Table 8 and Table 9 results are presented when all the spanning trees are generated. The calculation of the performance metrics are presented for backup paths that *can use* the spine edges in Table 8, and that *avoid* the spine edges in Table 9.

It is possible to observe in Tables 8 and 9 that for the polska network the performance metrics present equal values. The latter results are different when using uniform availabilities, as can be seen by comparing the results for polska network in Table 6 with the corresponding values in Table 7. This can be explained by observing that max $\min A_f$ in

Table 8 is very high when comparing with Table 6.

Moreover the results are similar when the backup path can use the spine (Table 8) or must avoid the spine (Table 9). This latter result can be explained by noting that a lexicographic version of the Dijkstra's algorithm is used, where the first objective function minimizes the hop count and the second objective function maximizes the backup path availability. When the availabilities are similar, which is the case for these networks, there is no significant difference for $A_{\mathcal{S}}$, due to the backup path being able to use the spine or having to avoid the spine.

The results when using the $c_l^{A'}$ as the edge weight and generating $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees to determine the spine are presented in Table 10. The last column of Table 10 (and in the Tables that follow), $\frac{k}{|\mathcal{N}||\mathcal{L}|}$, is the ratio between the order of the spanning tree corresponding to the bold value in the same line, and the total number of generated candidate spines.

To compare the results of the $c_l^{A'}$ cost a new version of the Case E, presented in section II, where the principal eigenvector of the adjacency matrix was used. In this case a weighted version of the adjacency matrix, Adj', is used where the element of the matrix, a'_{ij} where $i, j \in \mathcal{N}$, defined as:

$$a'_{ij} = \begin{cases} -\log(a_l), & \text{if } l \in \mathcal{L}. \\ 0, & \text{if } l \notin \mathcal{L} \end{cases} \tag{11}$$

Table 8: Results considering all the spanning trees. The backup path *can use* the spine edges.

Network	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f
polska	0.9998342	**0.9999999721**	0.2424	5.2727	2.6667	5	0.9999999033
	0.9998418	0.9999999692	0.2438	5.4545	2.6818	5	0.9999999045
	0.9998409	0.9999999714	0.2934	4.3636	3.2273	7	**0.9999999261**
nsf	0.9989943	**0.999998834**	0.2527	4.46154	3.2857	7	0.9999972794
	0.9989998	0.999998830	0.2511	4.6154	3.2637	7	0.999997147
	0.9987678	0.999998748	0.2815	4.3077	3.6593	8	**0.9999974322**
epan16	0.9996360	**0.9999998706**	0.2567	4.5333	3.85	9	0.9999996085
	0.9996743	0.9999998632	0.2233	4.9333	3.35	7	0.9999993615
	0.9996360	0.9999998706	0.2567	4.5333	3.85	9	**0.9999996085**

Table 9: Results considering all the spanning trees. The backup path *avoids* the spine edges.

Network	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f
	0.9998342	**0.9999999721**	0.2424	5.2727	2.6667	5	0.9999999033
polska	**0.9998418**	0.9999999692	0.2438	5.4545	2.6818	5	0.9999999045
	0.9998409	0.9999999714	0.2934	4.3636	3.2273	7	**0.9999999261**
	0.9989943	**0.9999988342**	0.2527	4.4615	3.2857	7	0.9999972794
nsf	**0.9989998**	0.999998830	0.2511	4.6154	3.2637	7	0.999997147
	0.9987678	0.999998748	0.2815	4.3077	3.6593	8	**0.9999974322**
	0.9996360	**0.9999998706**	0.2567	4.5333	3.85	9	0.9999996085
epan16	**0.9996743**	0.9999998632	0.2233	4.9333	3.35	7	0.9999993615
	0.9996360	0.9999998706	0.2567	4.5333	3.85	9	**0.9999996085**

The value of a_l is defined in equation (10). Also if edge $l \in \mathcal{L}$ connects nodes i and $j \in \mathcal{N}$. The components of the principal vector, v, of Adj' have a lower value for the nodes that belong to edges with higher availability. The cost of the edge l, $c_l^{E'}$, is defined as $c_l^{E'} = \frac{v_i + v_j}{2}$. In Table 11 the results of generating $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using the $c_l^{E'}$ as the edge weight are presented.

The results of the application of the costs $c_l^{A'}$ and $c_l^{E'}$, based on *weighted* metrics, are also compared with a *weighted* version of the metric $BC_k(l)$ where the edge cost are defined as: $c_l^{F'} = \frac{1}{BC_k(l)}$. Similar to the weighted eb_l metric in the $c_l^{A'}$ cost, in the $BC_k(l)$ metric the *weighted shortest paths* are used with the length of the shortest being replaced by the shortest past cost. Also, in this latter metric it was considered $k = 1$, which correspond to considering all the shortest paths with minimum cost plus the paths

Table 10: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using the $c_l^{A'}$ as edge weight. The backup path *can use* the spine edges.

Network	**Discarded spanning trees**	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f	$\frac{k}{\|\mathcal{N}\|\|\mathcal{L}\|}$
polska	0.8009	0.9998409	**0.9999999714**	0.2934	4.3636	3.2273	6	**0.9999999261**	0.2037
		0.9998418	0.9999999692	0.2438	5.4545	2.6818	5	0.9999999045	0.0417
nsf	0.8534	**0.9989998**	**0.9999988303**	0.2511	4.6154	3.2637	7	**0.9999971472**	0.3534
		0.9996452	**0.9999998627**	0.2528	4.5333	3.7917	9	0.9999994605	0.7772
epan16	0.9511	**0.9996601**	0.9999998618	0.2483	4.6667	3.725	9	0.9999994168	0.2120
		0.9995124	0.9999998396	0.3183	4.4	4.775	11	**0.9999996085**	0.8261
newyork	0.0	0.9937948	**0.9999594752**	0.1733	6.9333	2.6	4	**0.9998785935**	0.2207
		0.9937969	0.9999594298	0.1794	6.9333	2.6917	5	**0.9998785935**	0.2143
italia	0.0	**0.9998021**	**0.9999999526**	0.1500	5.5484	4.6512	9	0.9999998372	0.1291
		0.9997907	0.9999999514	0.1525	5.4839	4.7278	10	**0.9999998446**	0.5965
germany50	0.9657	0.9998363	**0.99999997**	0.1432	4.6122	7.0163	17	**0.9999998785**	0.6103
		0.9998374	0.9999999698	0.1392	4.6939	6.8204	16	0.9999998674	0.6393

Table 11: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using the weighted $c_l^{E'}$ as edge weight. The backup path *can use* the spine edges

Network	Discarded spanning trees	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f	$\frac{k}{\vert\mathcal{N}\vert\vert\mathcal{L}\vert}$
		0.9998342	**0.9999999721**	0.2424	5.2727	2.6667	5	0.9999999033	0.7269
polska	0.6296	**0.9998418**	0.9999999692	0.2438	5.4545	2.6818	5	0.9999999045	0.7361
		0.9998310	0.9999999717	0.2562	4.9091	2.8182	5	**0.9999999181**	0.9954
		0.9988804	**0.9999987848**	0.2604	4.4615	3.3846	7	0.9999968611	0.7406
nsf	0.6541	**0.9989784**	0.9999987119	0.2553	4.7692	3.3187	6	0.999994813	0.4624
		0.9988672	0.9999987613	0.2637	4.4615	3.4286	7	**0.9999971277**	0.7820
		0.9996060	**0.9999998468**	0.2439	4.6667	3.6583	8	0.9999995323	0.7745
epan16	0.7391	**0.9996125**	0.99999984	0.2406	4.6667	3.6083	8	0.9999993546	0.9103
		0.9995415	0.9999998335	0.2783	4.5333	4.175	11	**0.9999995803**	0.8859
		0.9919795	**0.9999506183**	0.1789	6.8	2.6833	5	0.9997622393	0.8571
newyork	0.0	**0.9919857**	0.9999493645	0.18	6.8	2.7	5	0.9997622393	0.4809
		0.9912422	0.9999485318	0.1917	5.8667	2.875	5	**0.999855272**	0.3661
italia	0.0	0.9996652	**0.9999999317**	0.1796	5.1613	5.5685	13	**0.9999997946**	0.2536
		0.9996653	0.9999999316	0.1856	5.0323	5.7540	14	0.9999997943	0.2758
germany50	0.6279	**0.9997941**	**0.9999999658**	0.1564	4.8571	7.6637	20	0.9999998459	0.9919
		0.9997906	0.9999999654	0.1606	4.8571	7.8702	21	**0.9999998539**	0.9690

with the next minimum cost. In Table 12 are presented the results of generating $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using the weighted $c_l^{F'}$ as edge weight.

It is possible to observe that comparing Table 11 with the results presented in Table 10 and in Table 12 that the number of discarded spanning trees is smaller when using the $c_l^{E'}$ costs.

By comparing the results presented in Tables 10-12 is possible to observe that the best $A_{\mathcal{S}}^{WP}$ is achieved by generating the k-minimum spanning trees with edges costs defined by $c_l^{F'}$. Nevertheless, there is one exception for the polska network where using the $c_l^{A'}$ or the $c_l^{E'}$ results in a higher $A_{\mathcal{S}}^{WP}$. For $A_{\mathcal{S}}$ the best results, with the exception of the polska

Table 12: Results considering $|\mathcal{N}|\,|\mathcal{L}|$ minimum spanning trees using $c_l^{F'}$ as edge weight. The backup path *can use* the spine edges

Network	Discarded spanning trees	$A_{\mathcal{S}}^{WP}$	$A_{\mathcal{S}}$	$eb_{\mathcal{S}}$	$ed_{\mathcal{S}}$	$h_{\mathcal{S}}$	$di_{\mathcal{S}}$	max min A_f	$\frac{k}{\vert\mathcal{N}\vert\vert\mathcal{L}\vert}$
polska	0.9815	**0.9996635**	**0.9999998562**	0.2528	4.5333	3.7917	9	0.9999993821	0.2639
		0.9996509	0.9999998396	0.2528	4.4	3.7917	8	**0.9999994038**	0.9815
nsf	0.9173	**0.9989998**	**0.9999988303**	0.2511	4.6154	3.2637	7	0.9999971472	0.6917
		0.9989920	0.999998826	0.2637	4.4615	3.4286	8	**0.9999973415**	0.9211
epan16	0.9837	**0.9996635**	**0.9999998562**	0.2528	4.5333	3.7917	9	0.9999993821	0.1549
		0.9996510	0.9999998396	0.2528	4.4	3.7917	8	**0.9999994038**	0.5761
newyork	0.0	**0.9938818**	**0.9999602687**	0.1661	7.8667	2.4917	4	**0.9998785935**	0.8610
italia	0.0	0.9998011	**0.9999999522**	0.1547	5.0968	4.7964	10	**0.9999998372**	0.6789
		0.9998064	0.9999999518	0.1552	4.9677	4.8125	10	0.9999997995	0.5679
germany50	0.9983	**0.9998396**	**0.9999999704**	0.1364	4.7347	6.6824	15	**0.9999998702**	0.7909

network, are achieved by using the $c_l^{A'}$ or the $c_l^{F'}$ as edges weights for the k-minimum spanning tree algorithm. As can be seen in Tables 10-12 the choice between the $c_l^{A'}$ or the $c_l^{F'}$ as edges weights, for generating the k-minimum spanning trees, for determining the spine with the best $A_{\mathcal{S}}$ depends on the network. Also, for both $c_l^{A'}$ and $c_l^{F'}$, the spine with best $A_{\mathcal{S}}$ also corresponds, in general, to a spine with a smaller diameter $di_{\mathcal{S}}$ than was obtained using $c_l^{E'}$. For the max $\min A_f$ performance metric it can be seen in Tables 10-12 that, with the exception of the nsf network, the larger values of the max $\min A_f$ are achieved by using the $c_l^{A'}$ as edge cost.

Another parameter that is important for the determination of the spine in large graphs is the fraction of generated trees, $\frac{k}{|\mathcal{N}||\mathcal{L}|}$, that are needed to achieve an admissible spine with best $A_{\mathcal{S}}$.

Observing the results in Tables 10-12 for $\frac{k}{|\mathcal{N}||\mathcal{L}|}$ it can be conclude that using the $c_l^{A'}$ cost the fraction of trees that are needed to achieve the spine with the best $A_{\mathcal{S}}$ is smaller. In short, as the most important metrics are $A_{\mathcal{S}}$ and $A_{\mathcal{S}}^{WP}$ its possible to say that using the $c_l^{F'}$ as a edge cost for generating the k-minimum spanning tree produces the best results, closely followed by $c_l^{A'}$.

Cost c_l^{A} with $\alpha = 1$ was used in subsection 4.3 with the objective of selecting for the spine edges with large edge betweenness. A more effective approach, to achieve that goal, would have been to calculate the spine that maximized the sum of eb_l, for all edges l in the spine. Such a spine could be obtained calculating the minimum spanning tree in a graph where the edge costs were given by $-eb_l + \max_l eb_l + 1$. A similar observation can be made regarding costs c_l^{E}, c_l^{F}, $c_l^{A'}$, $c_l^{E'}$ and $c_l^{F'}$. This is left for future work.

5 Conclusions

In this work it is discussed the spine concept and its is shown that by using a minimum spanning tree iterative algorithm in conjunction with a lexicographic Dijkstra algorithm a spine with a good $A_{\mathcal{S}}$ can be generated. It is demonstrated that calculating the backup path using a lexicographic Dijkstra algorithm, where the first cost is the hop count and the second the availability, produces better results than using just the minimization of the hop count, as would be expected. The comparison of the different edge costs, based on the graph topological metrics, demonstrates that the generating the spine using a cost, for the k-minimum spanning tree algorithm, based on the edge betweenness centrality produces better results regarding the considered performance measures of the spine. Based on this observation, three different edge costs were proposed, related to graph weighted metrics, for the generation of the spine. The first cost was based on the principal eigenvector of the adjacency matrix and the others on the weighted edge betweenness centrality. It was concluded that generating a spine using the cost based on the weighted k-betweenness centrality, $c_l^{F'}$, produces the best results.

References

[1] Abdulaziz Alashaikh, Teresa Gomes, and David Tipper. The spine concept for improving network availability. *Computer Networks*, 82(0):4 – 19, 2015. Robust and Fault-Tolerant Communication Networks.

[2] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology*, 2(1):113–120, January 1972.

[3] Stephen P. Borgatti and Martin G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466 – 484, 2006.

[4] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136 – 145, 2008.

[5] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk. A survey of resilience differentiation frameworks in communication networks. *Commun. Surveys Tuts.*, 9(4):32–55, October 2007.

[6] P. Cholda, J. Tapolcai, T. Cinkler, K. Wajda, and A Jajszczyk. Quality of resilience as a network reliability characterization tool. *IEEE Network*, 23(2):11–19, March 2009.

[7] Linton C. Freeman, Stephen P. Borgatti, and Douglas R. White. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141 – 154, 1991.

[8] H. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM Journal on Computing*, 6(1):139–150, 1977.

[9] T. Gomes, D. Tipper, and A. Alashaikh. A novel approach for ensuring high end-to-end availability: The spine concept. In *10th International Conference on Design of Reliable Communication Networks - DRCN 2014*, Ghent, Belgium, 1-3 April 2014.

[10] Teresa Gomes, David Tipper, and Abdulaziz Alashaikh. A novel approach for ensuring high end-to-end availability: The spine concept. In *Design of Reliable Communication Networks (DRCN), 2014 10th International Conference on the*, pages 1–8, April 2014.

[11] K. Jiang, D. Ediger, and D.A. Bader. Generalizing k-betweenness centrality using short paths and a parallel multithreaded implementation. In *Parallel Processing, 2009. ICPP '09. International Conference on*, pages 542–549, Sept 2009.

[12] N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding k minimum spanning trees. *SIAM Journal on Computing*, 10(2):247–255, 1981.

[13] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[14] M. Mezhoudi and C.-H. Kelvin Chu. Integrating optical transport quality, availability, and cost through reliability-based optical network design. *Bell Labs Technical Journal*, 11(3):91–104, 2006.

[15] W. Molisz and J. Rak. Quality of resilience in ip-based future internet communications. In *Transparent Optical Networks (ICTON), 2011 13th International Conference on*, pages 1–4, June 2011.

[16] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.

[17] M.E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54, 2005.

[18] J. Tapolcai, P. Cholda, T. Cinkler, K. Wajda, A Jajszczyk, A Autenrieth, S. Bodamer, D. Colle, Giuseppe Ferraris, H. Lonsethagen, I-E. Svinnset, and D. Verchere. Quality of resilience (qor): Nobel approach to the multi-service resilience characterization. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*, pages 1328–1337 Vol. 2, Oct 2005.

[19] David Tipper. Resilient network design: challenges and future directions. *Telecommunication Systems*, 56(1):5–16, 2014.

[20] Dongyun Zhou and S.S. Subramaniam. Survivability in optical networks. *Network, IEEE*, 14(6):16–23, Nov 2000.