Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees

Alan Kuhnle
Department of Computer &
Information Science & Engineering
University of Florida
Gainesville, Florida, USA
kuhnle@ufl.edu

Victoria G. Crawford
Department of Computer &
Information Science & Engineering
University of Florida
Gainesville, Florida, USA
vcrawford01@ufl.edu

My T. Thai
Department of Computer &
Information Science & Engineering
University of Florida
Gainesville, Florida, USA
mythai@ufl.edu

ABSTRACT

Motivated by networked systems in which the functionality of the network depends on vertices in the network being within a bounded distance T of each other, we study the length-bounded multicut problem: given a set of pairs, find a minimum-size set of edges whose removal ensures the distance between each pair exceeds T . We introduce the first algorithms for this problem capable of scaling to massive networks with billions of edges and nodes: three highly scalable algorithms with worst-case performance ratios. Furthermore, one of our algorithms is fully dynamic, capable of updating its solution upon incremental vertex / edge additions or removals from the network while maintaining its performance ratio. Finally, we show that unless $NP \subseteq BPP$, there is no polynomial-time, approximation algorithm with performance ratio better than $\Omega(T)$, which matches the ratio of our dynamic algorithm up to a constant factor.

KEYWORDS

Scalable algorithms; length-bounded multicut

ACM Reference Format:

Alan Kuhnle, Victoria G. Crawford, and My T. Thai. 2018. Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees. In *SIGMETRICS '18 Abstracts: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems Abstracts, June 18–22, 2018, Irvine, CA, USA*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3219617.3219650>

1 INTRODUCTION

Connectivity in a graph has historically been an important metric of the functionality of a network; that is, the service provided by the network is extant between two nodes if there exists a path between these two nodes. This consideration has led to the study of many forms of cutting problems in a network: *e.g.* the minimum cut problem, the minimum multicut problem and the sparsest cut problem, among many others [10].

Transcending connectivity, many network applications depend on some measure of network distance between a pair of connected

nodes. An edge weight representing a metric related to network functionality may be associated to each edge, and a pair of nodes only benefits from the network if the weighted, shortest-path distance between the pair is below a threshold T . For example, in an Industrial Internet-of-Things (IIoT) network [11], Quality-of-Service (QoS) metrics (*e.g.* packet loss, time delay) between a pair in the network are critical to many applications, and communication protocols have been developed to guarantee a threshold of QoS between communicating pairs [9]. As another example, consider a time-sensitive delivery on a road network, weighted by the travel time between destinations. Mere connectivity between a source and destination is insufficient when a guarantee on the delivery time is desired.

Consequently, a natural question is which edges or links in the network are critical for the network functionality between a desired set of pairs [4]. With this motivation, we consider the length-bounded multicut problem (LB MULTICUT), in which a weighted¹, directed graph G , threshold T , and set of pairs S is given, and the problem is to identify a minimum-size set of edges whose removal ensures the weighted, shortest-path distance between each pair in S is greater than the threshold T . Intuitively, the goal of this problem is to assess how robust the network is; the larger the size of the set of edges found, the more resilient the network is to perturbation in terms of edge failure; in addition, membership in this set of edges provides an indication of the importance of an edge to the desired functionality.

Modern networked systems are increasingly massive in scale [7], often with sizes of at least millions of vertices and edges. For example, one network of autonomous systems in the internet obtained from traceroutes has more than 12 million vertices and edges [6]. Furthermore, modern networks rapidly change as nodes and links enter and leave the network [6, 11–13]. Hence there is a need for dynamic algorithms that can efficiently update their solution in response to changes in the network, rather than recomputing from scratch on the modified network.

Contributions. Given an instance (G, d, S, T) of LB MULTICUT, let n be the number of vertices in G , m be the number of edges, and $T_0 = T/d_0$, in which d_0 is the minimum edge weight on G .

- We provide three highly scalable algorithms for LB MULTICUT: (1) SAP, a probabilistic, FPT-approximation algorithm with performance ratio $O(T_0 \log n)$ with probability at least $(1 - 1/m)$; (2) MIA, a multicut-based algorithm with performance ratio $O(Mn^{11/23})$, in which M is the number of directed

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGMETRICS '18 Abstracts, June 18–22, 2018, Irvine, CA, USA

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5846-0/18/06.

<https://doi.org/10.1145/3219617.3219650>

¹Each edge $e \in G$ has a positive weight $d(e)$.

multicuts employed by MIA; and (3) TAG, an approximation algorithm with performance ratio T_0 . Here, the *performance ratio* is the maximum ratio over all problem instances of the size of the set of edges returned by the algorithm divided by the optimal size.

- In the presence of dynamic edge / vertex insertions and deletions to the network, the dynamic version of TAG can efficiently update its solution in response to these incremental changes, while maintaining its performance guarantee of T_0 . Thus, our algorithm TAG is fully dynamic, which leads to massive speedups over static TAG.
- When T is fixed and edge weights are uniform, the previous best lower bound on the approximability of LB MULTICUT is $\Omega(\sqrt{T})$ [5] assuming the Unique Games Conjecture. We improve this lower bound to $\Omega(T)$ unless $NP \subseteq BPP$ ².
- We extensively evaluate our algorithms on large-scale, real-world networks. All three of our algorithms are demonstrated to scale to networks with billions of edges in under a few hours and to return nearly optimal solutions.

2 DEFINITIONS AND INAPPROXIMABILITY

In this section, we formally define the length-bounded multicut (LB MULTICUT) problem considered in this work.

PROBLEM 1 (MIN. LENGTH-BOUNDED MULTICUT (LB MULTICUT)). Given digraph G , positive weight function $d : E \rightarrow \mathbb{R}$, threshold T , and target set $S = \{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$, determine a minimum-size set S of edges such that $d(s_i, t_i) > T$ for all i after the removal of S from G , in which $d(s, t)$ is the d -weighted shortest paths distance from s to t . A problem instance may be represented by the tuple (G, d, S, T) .

Even in the case in which T is a fixed parameter and edge weights are uniform, LB MULTICUT is NP-hard as shown in Baier *et al.* [1] for the case when S consists of a single pair.

We show that the performance ratio of T_0 of TAG and the primal-dual algorithm are optimal up to a constant factor unless $NP \subseteq BPP$. The result holds for the special case of LB MULTICUT when T is a fixed parameter rather than part of the input; for clarity, we write T -LB MULTICUT to emphasize that T is a fixed constant. The proof of the theorem can be found in the full version of the paper [3]

THEOREM 1. Let $T \geq 16$. Unless $NP \subseteq BPP$, there is no polynomial-time algorithm to approximate T -LB MULTICUT within a factor of

$$\left\lceil \frac{T}{6} \right\rceil - 1 - \epsilon,$$

for any $\epsilon > 0$.

3 ALGORITHMS FOR LB MULTICUT

In this section, we present the algorithms SAP, TAG, and MIA for LB MULTICUT. Pseudocode and analysis of run time for each algorithm may be found in the full version of the paper [3]. In addition, the proofs of all theorems in this section may be found in the full version of the paper [3]. When the problem instance is clear from the context, OPT is defined to be the size of the optimal solution.

²Bounded-error probabilistic polynomial time: class of decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability bounded away from 1/2 [8].

3.1 Probabilistic, $O(T_0 \log n)$ FPT Approximation

In this section, we present SAP, an FPT-approximation algorithm for LB MULTICUT with ratio $O(T_0 \log n)$ with probability at least $1 - 1/m$. SAP runs in polynomial time when the parameter T_0 is fixed.

In essence, SAP is a greedy algorithm that attempts, in each iteration, to select an edge e that hits the largest number of paths in $\mathcal{P}$, where $\mathcal{P}$ is the set of T -bounded paths between pairs in S . Rather than an expensive listing of $\mathcal{P}$ to determine e , an estimator is employed by a path sampling procedure to select the best edge. This process is repeated iteratively until a feasible solution S is obtained; feasibility is checked by shortest-path computations.

THEOREM 2. With probability $1 - 1/m$, the solution S returned by SAP satisfies $O(T_0 \log n) \cdot OPT \geq |S|$.

3.2 Multicut Iterative Approach

In this section, we present another algorithm MIA to approximate LB MULTICUT with ratio $O(Mn^{11/23})$. In contrast to SAP, MIA is deterministic, and it is our only algorithm with a performance ratio that does not directly depend on T_0 . In the worst case, MIA may require superpolynomial time, but we found our implementation to be highly scalable in practice.

The general idea of MIA is as follows: until a feasible solution to the problem is produced, perform an approximate multicut on a subgraph of G , wherein the subgraph is constructed in order to ensure that the optimal multicut of S on this subgraph is a lower bound on the optimal solution. The solution produced is the union of these successive multicuts, and since each optimal multicut is a lower bound on OPT , the performance ratio is the number M of multicuts performed multiplied by the performance ratio of the approximation algorithm used for multicut.

THEOREM 3. Let (G, d, S, T) be an instance of LB MULTICUT. Then, MIA returns a feasible solution S within a factor $O(Mn^{11/23})$ of $OPT(G, d, S, T)$, in which M is the number of multicuts required.

3.3 Fully Dynamic T_0 -Competitive Algorithm

In this section, we present our T_0 -competitive Algorithm TAG that is capable of dynamically updating its solution to LB MULTICUT upon incremental changes to the network. After each change to the graph, TAG ensures that its solution maintains a worst-case, performance ratio of T_0 to the optimal solution on the updated problem instance; that is, TAG has a *competitive ratio* of T_0 .

In overview, TAG ensures that its solution S lies within the union U of the edges of a pairwise edge-disjoint collection of paths $\mathcal{U} \subseteq \mathcal{P}$, where $\mathcal{P}$ is the set of T -bounded paths between pairs in S ; since these paths are pairwise edge-disjoint, a worst-case guarantee of T_0 is enforced on U and therefore S since $S \subseteq U$. Internally, TAG monitors a set $W = U \setminus S$ of *pruned edges* that is disjoint from S . To enable its solution to be efficiently updated, TAG maintains a function $h : S \cup W \rightarrow \mathcal{U}$ that maps edges to the pairwise edge-disjoint set of paths $\mathcal{U}$.

Once a solution as described above is obtained, TAG maintains it by running TAG-ADD whenever an edge is added, and TAG-REMOVE whenever an edge is removed. To delete a vertex, all of its edges

may be deleted one by one and then the vertex may be deleted from the graph. Vertex addition may be handled similarly.

THEOREM 4. *Suppose we have graph G and a sequence of collections of edge additions or deletions $c_1, c_2, \dots, c_l$, which produces a sequence of graphs $G = G_0, G_1, \dots, G_l$. Then running TAG results in feasible solution S_i to LB MULTICUT on each G_i . If O_i is an optimal solution to LB MULTICUT on G_i , then $|S_i| \leq T_0|O_i|$.*

4 EXPERIMENTAL EVALUATION

We evaluate all of our approximation algorithms and demonstrate that in practice they return nearly optimal solutions and scale to networks with billions of nodes and edges. We compare the algorithms on static instances of LB MULTICUT, as well as evaluate the dynamic components of TAG. We summarize the key observations from the experimental results. The full experimental results and details on modifications to the algorithms evaluated may be found in the full version of the paper [3].

The algorithms were evaluated on both synthesized networks and traces of real networks. The synthesized networks we considered were small Erdos-Renyi (ER) graphs (with $n = 100$), in which each potential edge has probability p of appearing in the graph. On these small networks, we compare all algorithms with the optimal solution OPT. For real-world traces, we considered both weighted and unweighted topologies drawn from a variety of application domains. With the exception of RoadSF, all of these networks were collected by the Stanford Network Analysis Project [7]. Gnutella is a peer-to-peer network; Enron is an e-mail communication network, Google is a world-wide-web graph, Skitter is an autonomous-system graph obtained from the internet, and Friendster is a social network. Finally, RoadSF is the weighted, road network of San Francisco [2], which is normalized so that all distances lie in $[0, 10000]$.

4.1 Summary of results

In this section, we summarize the experimental results.

- Of our three algorithms, the solution qualities of MIA and TAG on our real-world traces were very similar; in terms of running time, MIA scales better with k , the number of pairs in the target set S . But in terms of memory usage, TAG scales better with k . The solution quality of SAP was worse than the other two, but usually by only a small factor. In addition, SAP usually ran the fastest, although for high k values it was occasionally beaten by MIA.
- As compared with static TAG, dynamic TAG demonstrated massive improvements in average running time of up to 10^8 for edge removal and 10^4 for edge addition and only lost small percentages of solution quality over millions of edge insertions and removals.
- The primal-dual algorithm PRIM consistently ran much faster than SAP, MIA, and static TAG, often by a factor of more than 10. However, its solution quality was usually worse by a factor of at least 5; on RoadSF, PRIM returned solutions nearly 1000 times larger than TAG.
- The FPT-approximation algorithm GEN performed well on the small ER networks with $n = 100$ but was unable to scale to larger networks due to both time and memory constraints stemming from the listing of all bounded paths required.

- On networks with very long path distances, SAP requires the parameter γ , a bias parameter in the sampling of SAP [3], to be set higher than 0.75 to obtain valid path samples. We observed this effect only on RoadSF, which had large distances corresponding to thousands of hops required for the sampling procedure.

5 ACKNOWLEDGEMENTS

The anonymous reviewers and Thang N. Dinh provided many helpful comments which improved the manuscript. This work was supported in part by NSF grants CNS-1443905 and EFRI 1441231, and DTRA grant HDTRA1-14-1-0055.

REFERENCES

- [1] Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Koehler, Petr Kolman, Ondrej Pangrac, Heiko Schilling, and Martin Skutella. 2010. Length-Bounded Cuts and Flows. *ACM Transactions on Algorithms* 7, 1 (2010), 1–27. <https://doi.org/10.1145/1868237.1868241>
- [2] Thomas Brinkhoff. 2000. Generating Network-Based Moving Objects. In *IEEE 12th International Conference on Scientific and Statistical Database Management*. 8–10.
- [3] Alan Kuhnle, Victoria G Crawford, and My T Thai. 2018. Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 1 (2018), 4.
- [4] A. Kuhnle, T. Pan, V.G. Crawford, M.A. Alim, and M.T. Thai. 2017. Pseudo-separation for assessment of structural vulnerability of a network. In *SIGMETRICS 2017 Abstracts - Proceedings of the 2017 ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems*. <https://doi.org/10.1145/3078505.3078538>
- [5] Euiwoong Lee. [n. d.]. Improved Hardness for Cut , Interdiction , and Firefighter Problems. *Arxiv preprint arxiv:1607.05133v1* ([n. d.]). [arXiv:arXiv:1607.05133v1](https://arxiv.org/abs/1607.05133v1)
- [6] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 177–187. <https://doi.org/10.11.1.59.8209>
- [7] Jure Leskovec and Andrej Krevl. 2014. {SNAP Datasets}: {Stanford} Large Network Dataset Collection. [url{http://snap.stanford.edu/data}](http://snap.stanford.edu/data).
- [8] Michael Sipser. 2006. *Introduction to the Theory of Computation* (second ed.). Thomson Course Technology.
- [9] Linus Thrybom and Gunnar Prytz. 2009. QoS in Switched Industrial Ethernet. *IEEE Conference on Emerging Technologies and Factory Automation* (2009). <https://doi.org/10.1109/ETFA.2009.5346995>
- [10] Vijay V Vazirani. 2003. *Approximation Algorithms* (first ed.). Springer-Verlag Berlin Heidelberg. <https://doi.org/10.1002/rsa.10038>
- [11] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics* 10, 4 (2014), 2233–2243. <https://doi.org/10.1109/TII.2014.2300753>
- [12] Qingchen Zhang, Chunsheng Zhu, Laurence T Yang, Zhikui Chen, Liang Zhao, and Peng Li. 2017. An Incremental CFS Algorithm for Clustering Large Data in Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 13, 3 (2017), 1193–1201. <https://doi.org/10.1109/TII.2017.2684807>
- [13] Chunsheng Zhu, Lei Shu, Takahiro Hara, Lei Wang, Shojiro Nishio, and Laurence T. Yang. 2014. A survey on communication and data management issues in mobile sensor networks. *Wireless Communications and Mobile Computing* 14 (2014), 19–36. <https://doi.org/10.1002/wcm.arXiv:arXiv:1405.1155v1>