Constructing vertex fault-tolerant spanners in polynomial-time

Faisal Mohammad[*], Caleb Robelle[†], and Ambrose G.Tuscano [‡]

Department of Computer Science, University of Maryland, Baltimore County

May 2020

Abstract

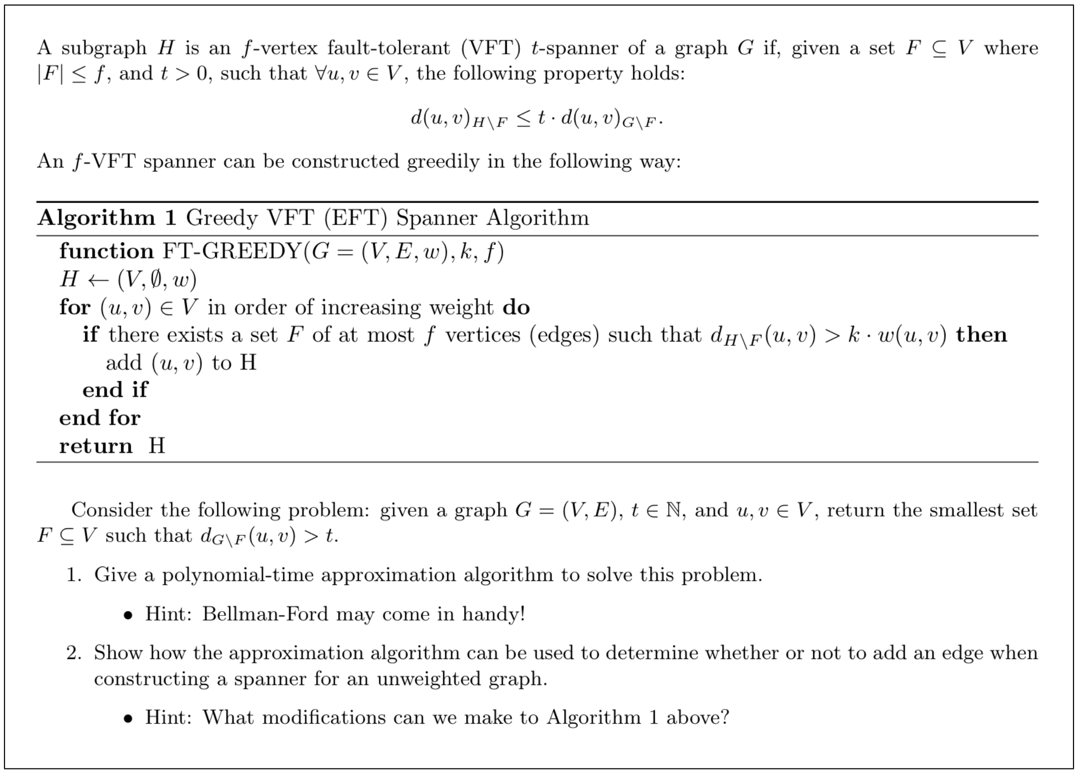
This report presents an approachable graph construction problem with regards to vertex fault tolerant (VFT) k-spanners. A k-spanner of a graph G is a sparse subgraph H which preserves distances up to a multiplicative constant k. An f-VFT k-spanner of G is a sparse subgraph H such that for any set of vertices F with $|F| \leq f$, $H \setminus F$ is a k-spanner of $G \setminus F$. Edge fault tolerant (EFT) spanners are defined similarly. Recent work has established optimal size bounds for fault tolerant spanners however, the standard algorithm for their construction takes time exponential in f. We provide a thorough explanation of the solution along with a discussion regarding the design and non-triviality of the problem. We conclude with a brief discussion regarding our homework problem design philosophy.

Keywords— Graph Spanners, Fault-Tolerant Spanners, Approximation Algorithms

[*]faisalm2@umbc.edu

[†]carobel1@umbc.edu

[‡]atuscan1@umbc.edu



A subgraph H is an f-vertex fault-tolerant (VFT) t-spanner of a graph G if, given a set $F \subseteq V$ where $|F| \leq f$, and $t > 0$, such that $\forall u, v \in V$, the following property holds:

$$d(u, v)_{H \setminus F} \leq t \cdot d(u, v)_{G \setminus F}.$$

An f-VFT spanner can be constructed greedily in the following way:

Algorithm 1 Greedy VFT (EFT) Spanner Algorithm

```
function FT-GREEDY($G = (V, E, w), k, f$)
$H \leftarrow (V, \emptyset, w)$
for $(u, v) \in V$ in order of increasing weight do
    if there exists a set $F$ of at most $f$ vertices (edges) such that $d_{H \setminus F}(u, v) > k \cdot w(u, v)$ then
        add $(u, v)$ to H
    end if
end for
return H
```

Consider the following problem: given a graph $G = (V, E)$, $t \in \mathbb{N}$, and $u, v \in V$, return the smallest set $F \subseteq V$ such that $d_{G \setminus F}(u, v) > t$.

1. Give a polynomial-time approximation algorithm to solve this problem.
    - Hint: Bellman-Ford may come in handy!
2. Show how the approximation algorithm can be used to determine whether or not to add an edge when constructing a spanner for an unweighted graph.
    - Hint: What modifications can we make to Algorithm 1 above?

Figure 1: Shown above is the problem statement as it would appear on a homework handout or textbook. Optionally, one could provide a qualitative discussion of spanners and fault-tolerance prior to the problem statement. However, this is not needed if we properly define the underlying data structures and their properties.

1 Introduction

Faisal Mohammad is a part-time PhD student at UMBC. He works as a full-time machine learning engineer at the Applied Physics Labs at Johns Hopkins University. Caleb Robelle is an undergraduate student majoring in computer science and mathematics. His research interests include: modeling circadian rhythms using systems of coupled ODEs; and geometric algorithms for counting roots of arbitrary polynomials over prime power rings. Ambrose Tuscano is a masters student in the computer science program. His research interest is mainly in machine learning for cybersecurity applications.

Bodwin and Patel's work, *A Trivial Yet Optimal Solution to Vertex Fault Tolerant Spanners*, presents an optimal upper bound on the worst-case size of fault-tolerant spanners [3]. The proof of the main result is quite involved and would not be fitting for an approachable homework problem. Instead, we invite students to understand the general definition of fault-tolerant spanners and use their preexisting knowledge on graph algorithms to modify an existing greedy approach. For this reason we present a brief overview of spanners before discussing the problem statement and solution in detail. Note that students attempting the homework problem only need the information provided in the problem statement shown in Figure 1 to solve the problem.

1.1 Graph Spanners

Given a graph $G(V, E)$ with a set of vertices V and edges E, a subgraph H is considered a *spanner* of G if it approximately preserves the shortest path metric of G up to a additive/multiplicative distortion factor. More formally, given a linear distortion function $h(x) = t \cdot x + \beta$ such that $t, \beta \geq 0$, H is a (t, β)-spanner of G if $\forall u, v \in V$:

$$d(u, v)_H \leq h(d(u, v)_G). \tag{1}$$

Where $d(x, y)$ is the path distance between Originally constructed in [6] [7], spanners have seen use cases in synchronizers, graph sparsifiers, and routing among others [7] [2] [4]. Optimizing size-stretch trade-offs is a key area of research, with several theoretical results having been made regarding upper bounds. The paper primarily concerns itself with t-spanners or $(t, 0)$-spanners with a multiplicative distortion factor t (likewise denoted as the stretch factor). The following greedy construction algorithm was first introduced by Althöfer et al [1].

Algorithm 1 Greedy Spanner Algorithm

```
function GREEDY($G = (V, E, w), k$)
$H \leftarrow (V, \emptyset, w)$
for $(u, v) \in V$ in order of increasing weight do
    if $d_H(u, v) > k \cdot w(u, v)$ then
        add $(u, v)$ to H
    end if
end for
return H
```

1.2 Fault Tolerant Spanners

Any real-life network-based system will naturally have links/edges that are susceptible to failure. A system that is fault-tolerant is designed to continue operating properly in the event of such failures. Levcopoulos, Narasimhan, and Smid [5] formalized such notion with regards to spanner by introducing fault tolerant spanners. Fault tolerant spanners have applications in a wide variety of settings; particularly in networks where the vertices or edges are prone to failure. In these settings, an efficient algorithm for constructing a fault tolerant spanner is necessary as the networks may be very large. We denote a subgraph H as an f vertex fault-tolerant $(t, 0)$-spanner of a graph G if, given a set $F \subseteq V$ such that $|F| \leq f$, $\forall u, v \in V$, the following property holds:

$$d(u,v)_{H\setminus F} \leq t \cdot d(u,v)_{G\setminus F}. \tag{2}$$

This property extends to edge fault-tolerant spanners as well. Algorithm 1 can intuitively be extended to construct VTF (EFT) spanners (Algorithm 2).

Algorithm 2 Greedy VFT (EFT) Spanner Algorithm

```
function FT-GREEDY($G = (V, E, w), k, f$)
H ← $(V, \emptyset, w)$
for $(u, v) \in V$ in order of increasing weight do
    if there exists a set $F$ of at most $f$ vertices (edges) such that $d_{H\setminus F}(u, v) > k \cdot w(u, v)$ then
        add $(u, v)$ to H
    end if
end for
return H
```

In a naive implementation, checking the *if* in Algorithm 2 condition takes exponential time in f. Finding a polynomial time algorithm for checking this condition would mean the previous greedy algorithm could now be implemented in polynomial time.

2 Problem Statement

The full problem statement can be found in Figure 1. For the sake of brevity, we present a shortened version given the fact we've covered graph spanners and the greedy spanner construction algorithm in the previous section. Ideally, those would be provided in a homework or problem set setting (Figure 1).

Consider the following problem known as the Length Bounded Cut problem (LBC). Given a graph $G = (V, E)$, $t \in \mathbb{N}$, and $u, v \in V$, return the smallest set $F \subseteq V$ such that $d_{G\setminus F}(u, v) > t$.

1. Give a polynomial-time approximation algorithm to solve this problem (Hint: the Bellman-Ford may come in handy!).
2. Show how the approximation algorithm can be used to determine whether or not to add an edge to when constructing a spanner for an unweighted graph (Hint: What modifications can we make to Algorithm 2?).

3 Model Solution and Correctness

Using the hints given in the problem statement, we break this section down into subsections discussing each parts of the problem. Afterwards, we provide brief proofs of correctness per part.

3.1 Solution to Part 1

We know that after t iterations, the Bellman-Ford algorithm will have calculated the shortest t-hop path from u to v. Each iteration of Bellman-Ford takes time $O(|E|)$ so we can determine if there is a path with at most t-hops from u to v in time $O(t \cdot |E|)$. Once such a path has been found we remove each node along the path from G and add them to our solution set F. We do this repeatedly until Bellman-Ford terminates without finding a path from u to v of at most t-hops. Since we remove at most 1 node each time we run Bellman-Ford, we get a running time of $O(t \cdot |E| \cdot |V|)$. Since we remove at most 1 node every time we run Bellman-Ford this gives us a running time of $O(t)$. Let L_{OPT} denote the smallest set required. Each path we remove from G must contain at least 1 node from L_{OPT} and since we remove up to t nodes from G for each node in L_{OPT} this means that $|F| \leq t \cdot |L_{OPT}|$. Therefore, we have a t-approximation algorithm for this problem which runs in polynomial time.

Algorithm 3 LBC Approximation Algorithm

```
function LBC-APPROX($G = (V, E, w), u, v, t$)
$L \leftarrow \emptyset$
while there exists a path $P$ from $u$ to $v$ with at most $t$ hops do
    $L \leftarrow P$
    $V = V \setminus P$
end while
return L
```

3.2 Solution to Part 2

We can use this approximation algorithm to determine if there is a fault set in H which requires us to add a certain edge. When considering an edge (u, v) run the approximation algorithm on input H, nodes u, v and let $t = d(u, v)$. If the size of F is greater than kf we add the edge to the spanner. Otherwise we leave it out.

Algorithm 4 Modified Greedy VFT Spanner Algorithm

```
function FT-GREEDY($G = (V, E), k, f$)
$H \leftarrow (V, \emptyset, w)$
for all $\{u, v\} \in E$ in order of increasing weight do
    Let $\ell$ be the value returned by our approximation algorithm on input $H$, length $k$, and nodes $u, v$.
    if $\ell \leq kf$ then
        add $(u, v)$ to H
    end if
end for
return H
```

3.3 Correctness of the Solution to Part 1

We claim that this is a t-approximation algorithm to the problem.

Proof. Let L_{OPT} denote the optimal solution to a given instance of LBC: $(G = (V, E), u \in V, v \in V, t \in \mathbb{N})$. We know that each path between u and v with at most t-hops must contain at least 1 node from L_{OPT}. We also have that each of the paths we remove from G are disjoint. Each time we remove a path between u and v, we add at least one element of OPT and no more then $t - 1$ nodes to L. Therefore, once our algorithm is finished, we must have that $|F| \leq (t - 1) \cdot |OPT| \leq t \cdot |OPT|$.

□

It is easy to show that this algorithm runs in polynomial time relative to the size of the input graph.

Proof. Running t iterations of Bellman-Ford on a graph $G(V, E)$ takes time $O(|E| \cdot t)$. Since we could possibly remove only 1 node after each round of Bellman-Ford, we run Bellman-Ford at most $|V|$ times giving us a complexity of $O(|V| \cdot |E| \cdot t)$

□

3.4 Correctness of the Solution to Part 2

Proof. Let, $(u, v) \in E$ and $F \subseteq V$ with $|F| \leq f$ and $u, v \notin F$. We have two possible cases: either $(u, v) \in E(H)$, or $(u, v) \notin E(H)$. In the first case we have that $(u, v) \notin E(H)$. Therefore $d(u, v)_{H \setminus F} \leq t \cdot d(u, v)_{G \setminus F}$. In the second case we know that when (u, v) was considered, the set L returned by LBC-APPROX had size greater than $k \cdot f$ otherwise (u, v) would have been added to H. This means that $f < |L_{OPT}|$ which means that at that stage, there was no fault set of f or fewer nodes such that $d(u, v)_{H \setminus F} > t \cdot d(u, v)_{G \setminus F}$. We must have that $d(u, v)_{H \setminus F} \leq t \cdot d(u, v)_{G \setminus F}$. Therefore H is an f-VFT t-spanner of G.

□

In the modified greedy algorithm, we run LBC-APPROX and perform several constant time operations E times. Giving us a time complexity of $O(|V| \cdot |E|^2 \cdot t)$.

4 Discussion - Strategy and Design

Alongside the technical efforts of formulating a problem and solution included: thinking of a strategy from a student's perspective; and pinpointing our philosophy of what constitutes a homework problem. We conclude with a discussion on the various lessons learned for this task.

4.1 Recommended Strategy to Digest the Problem

We don't expect graduate students to immediately know what spanners are. Therefore, we encourage students to draw out the graph structure and visualize the greedy construction of the spanner for a couple of steps. Ideally this should be done before using the hints given. For Part 1 of the problem statement, one must consider what information is available after each iteration of the Bellman-Ford algorithm. How can such information be utilized to determine what nodes to remove such that the distance between nodes u, and v greater than t? For Part 2, students should examine the provided greedy spanner construction algorithm and attempt to pinpoint where in the pseudocode can one make modifications. By analyzing the **if** condition in the loop, students can hopefully deduce how expensive this check is.

4.2 Homework Problem Design Philosophy

An excellent homework problem is one which students utilize their existing knowledge of algorithms to draw basic yet nontrivial clues from a rigorous paper. Famous problems that appear in algorithm textbooks have been effectively condensed to focus on a key idea of a theoretical result. For our problem, we assumed: basic knowledge of graph theory and algorithms; complexity analysis; approximation algorithms; and mathematical proof techniques. We do not assume any prior knowledge of: graphs spanners; fault tolerance; or the length-bounded cut problem. The problem is self-contained. The student should (with the information provided by us) be able to solve problem and would not need to research more into details of the problem. We provide just enough hints to keep the problem non-trivial with a solution that requires critical thinking (Figure 2).

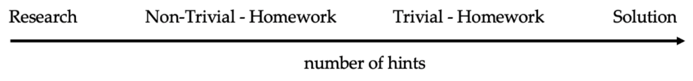


Figure 2: A qualitative figure providing how the transformation between a research problem to a solution is dependent on the number of hints provided. This is of course not drawn to scale as the leap from a research problem to a non-trivial homework problem is the largest.

4.3 Lessons Learned

Creating a homework problem from a research paper was a challenging yet rewarding experience. We were required to draw what we already know about graph theory and algorithms to decipher the paper to the best of our ability. Of course this wasn't easy as the paper was a specialized topic on spanners. Reading the paper helped us appreciate the level of intricacy and effort required to have such research published at top conferences. After making sense of the contents of the paper, the homework design process began. This demanded a lot of brainstorming and sharing ideas between the team. The real challenge was to improve on the abstract idea that we had and work up a solution to that. We had to careful and use ourselves as the upper bound on the knowledge required to solve the problem. Unlike a student given the problem for the first time, we had a whole semester to formulate it. Therefore, it needed to be simple enough that a student who's never heard of spanners or fault-tolerance can still solve it in 3-6 hours. This project also allowed us to appreciate professors' and book authors' ability to break such complex theoretical results down to the homework problems we see in our textbooks.

5 Acknowledgements

Our homework problem and reference paper were selected due to Caleb's research interests in graph spanners. Therefore Caleb served as the technical guide throughout the project and played a main role in designing the problem and solution. As the only PhD student of the group, Faisal was the project manager of the group. He contributed significantly to each of the deliverables, including writing content, proofreading and submissions. Ambrose played a major role in designing the slides and organizing the presentations. His main tasks involved fleshing out the design philosophy of the homework problem and contributing to each of the deliverables as well. Overall, everyone contributed in different ways and put an equal amount of effort. Conclusively, we'd like to thank our professor Dr. Alan Sherman for providing feedback on the initial draft as well as steering us to meet the appropriate requirements for a homework problem based off a research paper.

Appendix A: Research conference and papers

References

[1] Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[2] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 514–522 vol.2, 1990.

[3] Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). *CoRR*, abs/1710.03164, 2017.

[4] Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. January 2012.

[5] Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 186–195, New York, NY, USA, 1998. Association for Computing Machinery.

[6] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[7] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, July 1989.