Sparse Fault Tolerant BFS Trees

Gengyan Li

April 14, 2018

Introduction

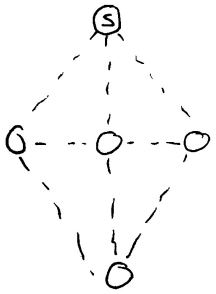
We begin by first clarifying what is meant by Sparse Fault-Tolerant BFS Trees.

In this context, a Breadth-First Search (BFS) Tree, is defined as a subset of edges of a connected graph that contains the shortest paths between a predefined source node and all other nodes. Though strictly speaking not always an actual Tree, we will still refer to it as such for convenience

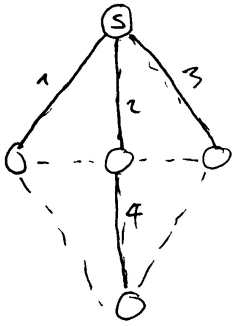
A Fault tolerant BFS Tree then, informally is a subset of edges such that even if a single (and only a single) edge is removed from the graph, the resulting tree is still a BFS tree with regard to the modified graph. More precisely, the tree must contain the BFS tree with respect to any subset of the graph containing all but one edge.

Sparsity refers to finding the minimal possible number of edges for such a FT (Fault-Tolerant) BFS Tree.

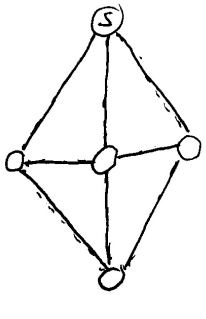
As such, our goal is the find a sparse (not necessarily sparsest) FT-BFS Tree in polynomial time.



(a) A very simple graph



(b) The BFS Tree generated from it



(c) The most simple FT BFS Tree (T = G)

Definitions

We begin by formally defining the actual problem that we attempt to solve.

Definition 1. *A graph T is an* FT-BFS *tree for G w.r.t. a source node $s \in V$, iff for every edge $e \in E$ and for every $v \in V$, $\texttt{dist}(s, v, T \backslash \{e\}) = \texttt{dist}(s, v, G \backslash \{e\})$.*

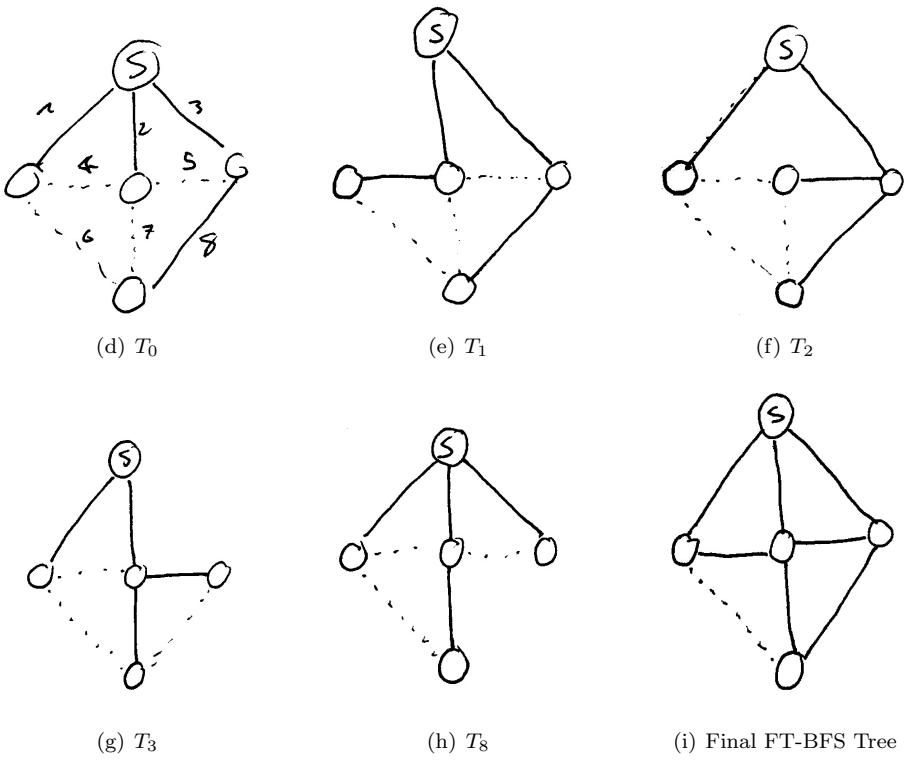
We first provide an algorithm that constructs an FT-BFS Tree with $O(n^{3/2})$ edges, and then show that the lower bounds is actually the same, $\Omega(n^{3/2})$, in worst case scenarios.

Upper Bound and algorithm

We present an algorithm that constructs FT-BFS trees and also provide an upper bounds on the number of edges it has, as well as show that it runs in polynomial time.

Theorem 1. *There exists a polynomial time algorithm that for every n-vertex graph G and source node s constructs an* FT-BFS *tree rooted at s with $O(n^{3/2})$ edges.*

The Algorithm. First, we introduce some symbolic perturbation to the edge lengths, such that there is always exactly one shortest path to each vertex. Let T_0 be the BFS tree rooted at s in G, computed according to the weight assignment. For every $e_j \in T_0$, compute the BFS tree rooted at s in $G \backslash \{e_j\}$. The final FT-BFS Tree is the union of all such trees, denoted as T^*.



(d) T_0 (e) T_1 (f) T_2

(g) T_3 (h) T_8 (i) Final FT-BFS Tree

Figure 1: An example use of the algorithm

We provide an example below, using the same graph as in the introduction. We define the shortest paths in ties as the one that has $\texttt{max}(\underset{e_j \in P}{\texttt{min}}(j))$

First, we construct the BFS Tree T_0 given in Figure 4 a. Next, we construct the trees T_j where $e_j \in T_0$, by

removing an edge in T_0 from the Graph G and constructing a BFS Tree from the resulting graph, as seen in Figure 4 b to e. Finally, we take the union of all constructed trees, and obtain the tree in Figure 4 f.

This algorithm is extremely simple, but we can still prove that it provides a very sparse FT-BFS tree in polynomial time.

Observation 1. *$T^*(s)$ is an FT-BFS tree.*

The proof for this is trivial, and follows from the construction of $T^*(s)$.

Time Analysis.

First, the construction of T_0 occurs in at most $O(n+m)$ time, where m is the number of edges in G, being a simple BFS Tree construction. This tree has n edges (as it is a BFS Tree), and as such n BFS trees $T_{1..n}$ must also be constructed. Thus, the whole algorithm occurs in $O(n(n+m))$ time, which is equivalent to $O(nm)$ time, as $n < m$ (as a property of connected graphs). As $m \leq n^2$, we conclude that the algorithm occurs in polynomial time.

Size Analysis. To prove Theorem 1, we need to show that the size of the constructed Tree is $O(n\sqrt{n})$.

We will prove this by showing that for every vertex, at most $O(\sqrt{n})$ neighboring edges will be added to the final tree.

We note that every edge in T^* is either part of the simple BFS Tree T_0, or the **last edge in some replacement path to some vertex** for some T_i with $e_i \in T_0$. We know this as if an edge is not in T_0, then it must be part of the shortest path in the tree T_i to one of the vertices it's connected to, by definition of BFS Trees. As the number of edges in T_0 is bounded by $O(n)$, only the number of the latter kind of edges are important.

As we know each edge is the last edge in some path to some vertex u, we then focus on only one such vertex u in G, and the mentioned last edges neighboring u.

We note that if an edge neighboring u is in T^* but not in T_0, then the tree we obtained that edge from must have been constructed by removing an edge from the path from the source to u, as otherwise the original path would be the shortest (as we've defined the structure to have no equally long paths), and we need no additional path for that tree. That is, any such edge is part of the replacement path for some removed edge in the path $s-u$.

For example, in the figure below, it does not matter what other edges are removed; only removing edges from the path to u will create a replacement path to u.

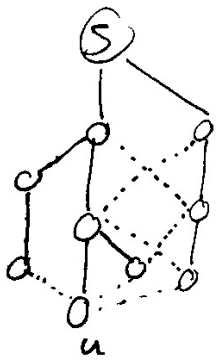


Figure 2: Possible new edges for u

We now consider only one replacement path for every new edge added that is adjacent to u.

We then observe that additionally, any replacement path will not touch the original path after it diverges from it. That is, any replacement path will be a subpath of the direct path from s to u, followed by a path that is vertex disjunct from that path. Consider the removed edge e by which the replcement path was generated. As by definition of BFS Trees every subpath of a path is optimal, if it did touch at a given vertex v after divergence from the path at some vertex w, then either the removed edge was between w and v, or v and u. In either case, the subpath between the other two nodes for the replacement path would not be the shortest optimal path, contradicting our optimality assumption.

We also observe that any two replacement paths to the node u do not share any vertices except for the direct path in the simple BFS Tree T_0. This can be shown similarly to the previous observation. As the two paths end in different edges (by our consideration of only replacement paths ending in different edges), if they shared some node v, there would be two different shortest paths from v to u, contradicting our assumption of there being one optimal path.

From those observations, and due to the fact that the replacement paths necessarily contain more vertices than the original path (due to optimality), we can then conclude that each replacement path adds at least $k-1$ vertices, where k is the number of edges away from u that the replacement path diverged from the original path from s to u. We also observe that k is a unique natural number for each replacement path, as we would again have the issue of optimality being contradicted by two replacement paths starting from the same vertex.

Now, we now can prove that there are less than or equal to $\sqrt{2n}$ new edges adjacent to u. suppose we have l new edges with $l > \sqrt{2n}$. Then, we have l vertices from s to u. We also have, for each replacement path, at least $k-1$ new vertices. As each k is a unique natural number, and we have l different replacement paths, we get

$$n \geq l + \sum_{k=1}^{l} k - 1 = \sum_{k=0}^{l} k = \frac{l^2 + l}{2} > \frac{\sqrt{2n}^2}{2} = n$$

which is a clear contradiction.

As we only have n possible u's, we've successfully shown that there are at most $O(n^{3/2})$ edges in the tree T^*, proving the theorem.

Lower bounds

We then present a lower bound on the number of edges required for an FT-BFS tree with n vertices, in the worst case, which happens to have the same asymptotic complexity as our algorithm.

Theorem 2. *There exists an n-vertex graph $G(V, E)$ and a source node $s \in V$ such that any* FT-BFS *tree rooted at s has $\Omega(n^{3/2})$ edges.*

Proof : We prove this by constructing for any sufficiently large n an n-vertex graph for which any FT-BFS Tree requires at least $\Omega(n^{3/2})$ edges.

For any given n, let

$$d = \lfloor \sqrt{n}/2 \rfloor$$

The graph consists of four main components.

First we construct a path $\pi = [s = v_1, \ldots, v_{d+1} = v^*]$ of length d.

Then, we construct a node set $Z = \{z_1, \ldots, z_d\}$ and connect each of them with a path of decreasing length, $P_1, \ldots, P_d$, to a corresponding vertex v_j. The length of each path is

$$|P_j| = 4 + 2(d - j)$$

including end nodes. Altogether, the set of nodes in these paths, $Q = \bigcup_{j=1}^{d} V(P_j)$, is of size

$$|Q| = \sum_{j=1}^{d} |P_j| = \sum_{j=1}^{d} (4 + 2d - 2j) = 4d + 2d^2 - \frac{2(d^2 - d)}{2} = d^2 + 5d$$

as no two Ps intersect.

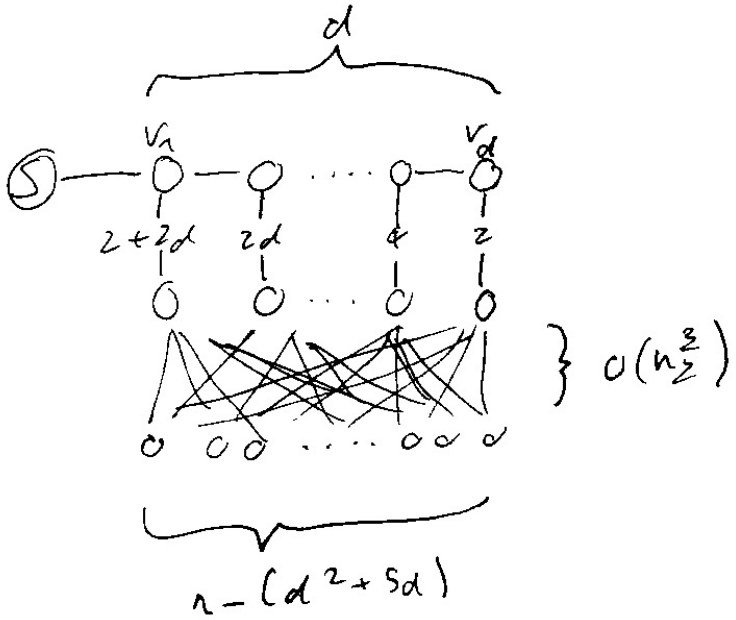


Figure 3: The first and second components of the graph.

The third component is a set of nodes X of size $n - (d^2 + 5d)$ (all remaining nodes). The last component is a complete bipartite graph $B = (X, Z, \hat{E})$ connecting X to Z.

Note that

$$(|Q| = \lfloor \sqrt{n}/2 \rfloor^2 + 5\lfloor \sqrt{n}/2 \rfloor) \leq n/2$$

for sufficiently large n. Consequently,

$$|X| = n - |Q| \geq n/2,$$

and

$$|\hat{E}| = |Z| \cdot |X| \geq n/2 \cdot \lfloor \sqrt{n}/2 \rfloor \geq n^{3/2}/8$$

(again, for sufficiently large n).

We now show that every FT-BFS tree constructed from G must contain all the edges $\hat{E}$ (the blue trapezoid in the figure).

Assume, towards contradiction, that there exists an FT-BFS tree that does not contain $e_{i,j}$. Note that upon the failure of the edge $e_j = (v_j, v_{j+1}) \in \pi$, the unique $s - x_i$ shortest path connecting s and x_i in $G \backslash \{e_j\}$ is $P'_j = \pi[v_1, v_j] \circ P_j \circ [z_j, x_i]$, and all other alternatives are strictly longer. Since $e_{i,j} \notin T'$, also $P'_j \not\subseteq T'$, and therefore $dist(s, x_i, G \backslash \{e_j\}) < dist(s, x_i, T' \backslash \{e_j\})$, in contradiction to the fact that T' is an FT-BFS tree. It follows that every FT-BFS tree T' must contain at least $|\hat{E}| = \Omega(n^{3/2})$ edges. The theorem follows. □

References

[1] M. Parter, D. Peleg, *Sparse Fault-Tolerant BFS Trees*, ESA 2013.