Sparse Weight Tolerant Subgraph for Single Source Shortest Path

Diptarka Chakraborty
Computer Science Institute of Charles University, Prague, Czech Republic
diptarka@iuuk.mff.cuni.cz

Debarati Das
Computer Science Institute of Charles University, Prague, Czech Republic
debaratix710@gmail.com

Abstract

In this paper we address the problem of computing a sparse subgraph of any weighted directed graph such that the exact distances from a designated source vertex to all other vertices are preserved under bounded weight increment. Finding a small sized subgraph that preserves distances between any pair of vertices is a well studied problem. Since in the real world any network is prone to failures, it is natural to study the fault tolerant version of the above problem. Unfortunately, it turns out that there may not always exist such a sparse subgraph even under single edge failure [Demetrescu *et al.* '08]. However in real applications it is not always the case that a link (edge) in a network becomes completely faulty. Instead, it can happen that some links become more congested which can be captured by increasing weight on the corresponding edges. Thus it makes sense to try to construct a sparse distance preserving subgraph under the above weight increment model where total increase in weight in the whole network (graph) is bounded by some parameter k. To the best of our knowledge this problem has not been studied so far.

In this paper we show that given any weighted directed graph with n vertices and a source vertex, one can construct a subgraph of size at most $e\cdot(k-1)!2^k n$ such that it preserves distances between the source and all other vertices as long as the total weight increment is bounded by k and we are allowed to only have integer valued (can be negative) weight on edges and also weight of an edge can only be increased by some positive integer. Next we show a lower bound of $c\cdot 2^k n$, for some constant $c \geq 5/4$, on the size of such a subgraph. We further argue that the restrictions of integral weight and integral weight increment are actually essential by showing that if we remove any one of these two we may need to store $\Omega(n^2)$ edges to preserve the distances.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases Shortest path, fault tolerant, distance preserver, graph algorithm

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.15

Funding The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement no. 616787.

Acknowledgements The first author would like to thank Pavan Aduri and Vinodchandran N. Variyam for some helpful discussions during initial phase of this work and a special thank to Pavan Aduri for suggesting to study this problem. Authors also thank Keerti Choudhary, Shahbaz Khan and Michal Koucký for many valuable suggestions and comments.

1 Introduction

In the real world, networks are prone to failures and most of the time such failures are unpredictable as well as unavoidable in any physical system such as communication network or road network. For this reason, in the recent past, researchers study many graph theoretic questions like connectivity [29, 27, 4, 23], finding shortest distance [18], building data structure that preserves approximate distances [25, 16, 14, 19, 7, 10, 3] etc. under the fault tolerant model. Normally such failures are much smaller in number comparative to the size of the graph. Thus we can associate a parameter to capture the number of edge or vertex failures and try to build fault tolerant data structures of size depending on this failure parameter for various graph theoretic problems.

Unfortunately, in case of single source shortest path problem, it is already known from [18] that there are graphs with n vertices for which to preserve the distances under even single edge failure, we need to store a subgraph of size at least $\Omega(n^2)$. On the other hand, in case of reachability problem we know a construction of connectivity preserving subgraph of size only $O(2^k n)$ [4] where k is the number of edge failures. However, in the real world it is not always the case that there are failures of edges or vertices. Instead, for weighted graphs, weight of any edge or vertex can be increased. For simplicity, we consider weight to be only on the edges. In general, weight of an edge captures aspect like congestion on a particular link in a network. So it is quite natural to consider the scenario when some links (edges) become more congested. Again the good thing is that most of the time such congestion is bounded, i.e., over a network total increase in congestion is bounded because of many reasons like bounded maximum number of consumers present in a network at any particular time. One can easily capture the increase in congestion by a parameter k that bounds the amount of increase in weight of edges over the whole graph. Occurrence of such bounded congestion motivates us to study the single source shortest path problem under this model.

In this paper we initiate the study of single source shortest path problem for weighted directed graphs in the bounded weight increment model. The main goal is to find a sparse subgraph that preserves the (shortest) distance between a designated source and any other vertex under weight increment. We formally define such a subgraph below.

▶ **Definition 1** (k-WTSS)**.** Given a graph G along with a weight function w, a source vertex $s \in V(G)$ and an integer $k \geq 1$, a subgraph $H = (V(G), E')$ where $E' \subseteq E(G)$ is said to be a *k-Weight Tolerant single source Shortest-path Subgraph (k-WTSS)* of G if for any weight increment function $I : E(G) \to \mathbb{N}$ such that $\sum_{e \in E(G)} I(e) \leq k$, the following holds: for the weight function defined by $w'(e) = w(e) + I(e)$ for all $e \in E(G)$,

$$dist_{G,w'}(s,t) = dist_{H,w'}(s,t) \text{ for any } t \in V(G).$$

Though in the above definition we restrict ourselves to an increment function whose range is $\mathbb{N}$, one can naturally extend the definition to any range of increment functions. However, if we take the range to be rational numbers then there may not exist any sparse k-WTSS even for $k = 1$ (see Section 6). This is the reason why we consider such restriction on increment function in the above definition.

Single source shortest path is one of the most fundamental problems in Graph Theory as well as in computer science. Thus construction of a sparse k-WTSS is interesting from both the theoretical and practical point of view. One can also view the problem of finding k-WTSS as a generalization of finding k-Fault Tolerant (single source) Reachability Subgraph (k-FTRS) for which optimal solution is known due to [4]. If in the given graph one assigns zero weight on the edges, then any k-WTSS of that graph will also be a k-FTRS. This is because we can view each edge fault as incrementing weight by one and then it is easy to see that for any vertex t there exists an $s-t$ path iff the shortest distance between s and t is zero under this reduction. This fact also motivates the study of constructing k-WTSS because k-FTRS has several applications like fault tolerant strong-connectedness [4], dominators [24, 4], double dominators [34] etc. We have already mentioned that it is possible to represent congestion in any network by incrementing weight of the links. Unlike to the edge faults, which can be thought of as an independent and equally likely process, congestion is strongly time-variant (e.g. [33]). For example, if at any point of time some edge is congested and it has small out-degree then at the next time step some or all of its out edges will also be congested. However the good thing about k-WTSS subgraph is that no matter how the congestion occurs or propagates, as long as the total weight increment is bounded by k, a k-WTSS subgraph will preserve the distances. Hence for the practical purposes it is always useful to construct (and store) a sparse k-WTSS (if exists). In spite of being an appealing problem, to the best of our knowledge the problem of constructing sparse k-WTSS has not yet been studied. Although, a lot of research has been done on different versions of shortest path problem under *dynamic* weight changes [32, 30, 17, 12] which have found several applications in network optimization, internet routing, databases and many more.

The main contribution of this paper is to provide an efficient construction of a sparse k-WTSS for any $k \geq 1$, where sparsity of k-WTSS depends on the parameter k.

▶ **Theorem 2.** *There exists an $O((k)^k m^2 n)$-time algorithm that for any given integer $k \geq 1$, and a given directed graph G with n vertices and m edges along with a weight function $w : E(G) \to \mathbb{Z}$ and a source vertex s, constructs a k-WTSS of G with at most $O_k(n)$ edges where $O_k(\cdot)$ notation denotes involvement of a constant that depends only on the value of k. Moreover, in-degree of every vertex in this k-WTSS will be bounded by $e \cdot (k-1)!2^k$.*

Next, we prove a lower bound of $c \cdot 2^k n$ for some constant $c \geq 5/4$, on the size of k-WTSS.

▶ **Theorem 3.** *For any positive integer $k \geq 2$, there exists a positive integer n_k so that for all $n > n_k$, there exists a directed graph G with n vertices and a weight function $w : E(G) \to \mathbb{Z}$, such that its k-WTSS must contain $c \cdot 2^k n$ many edges for some constant $c \geq 5/4$.*

We provide the proof of Theorem 3 in the full version [13]. Note that as we have previously argued that the construction of k-WTSS implies a construction of k-FTRS, so $2^k n$ lower bound on the size of a k-FTRS due to [4] also directly gives the same lower bound on size of a k-WTSS. In the above theorem we slightly strengthen that lower bound by a constant factor for our problem. We also show that considering rational valued weight function or rational valued weight increment function (instead of integer valued as in the above two theorems) makes the problem of finding sparse k-WTSS impossible. More specifically, we show that in both the cases there are graphs with n vertices for which any k-WTSS must be of size at least $\Omega(n^2)$ even for $k = 1$ (see Section 6).

One can further relax our model by also allowing decrement operation on edge weight. Weight decrement is also natural in real life applications because for any network it is possible that some links become less congested. Unfortunately, one can easily show that there are graphs for which there is no sub-quadratic sized subgraph that preserves the distances from a single source under this relaxed model. Readers may refer to the full version [13] for the details.

Related works

Single source shortest path is a well studied problem under the edge or vertex failure model. Similar to our definition of k-WTSS, one can easily define k-Fault Tolerant Shortest-path

Subgraph (k-FTSS) that preserves the distance information from a specific source vertex under at most k edge failures. Unfortunately, we know that there are weighted graphs for which no sparse k-FTSS exists even for $k = 1$, i.e., there are weighted graphs with n vertices for which any 1-FTSS must contain $\Omega(n^2)$ many edges [18]. This lower bound on size of 1-FTSS is even true for undirected graphs. However, better bounds are known for unweighted graphs for $k \leq 2$. Parter and Peleg [28] provided a construction of $O(n^{3/2})$ sized 1-FTSS and showed that this bound is optimal. Later, Parter [27] extended the construction to the case $k = 2$ for undirected graphs on the cost of weakening the bound and gave an algorithm to compute 2-FTSS of size $O(n^{5/3})$ along with a matching lower bound. This result has very recently been extended to directed graphs [23]. For general k, a construction of sub-quadratic sized FTSS is known for undirected unweighted graphs due to [10].

However, the situation is much better for single source reachability problem which is closely related to single source shortest path problem. Baswana *et al.* [4] showed that we can compute k-FTRS, which is a subgraph that preserves the reachability information from a given source under at most k edge failures, containing $2^k n$ many edges. They also provided a matching lower bound. We have already argued that computing k-FTRS can be reduced to computing k-WTSS and thus it is natural to ask whether a similar result also holds for k-WTSS. Another interesting related problem is to compute fault tolerant reachability oracle. It is trivial to see that using $O(2^k n)$ size k-FTRS [4] one can answer any reachability query in $O(n)$ time for any constant value of k. However for $k \leq 2$, $O(n)$ size data structure is known that can answer any single source reachability query in $O(1)$ time [24, 15].

Coming back to the shortest path problem, instead of preserving the exact distances (between any pair of vertices), if we consider to preserve the distances only approximately, then much better results are known. Such approximate distance preserving subgraphs are called *spanners*. Construction of spanners with both additive and multiplicative stretch have been studied extensively [5, 37, 1, 2]. Fault tolerant version of spanners were first introduced in the geometric setting [25]. For k edge failures, construction of a $(2l - 1)$ multiplicative spanner of size $\widetilde{O}(kn^{1+1/l})$, for any $k, l \geq 1$, was provided in [14] whereas for k vertex failures, upper bound on size is known to be $\widetilde{O}(k^{2-1/l} n^{1+1/l})$ [19]. In case of single edge failure, for single source undirected graphs, construction of a $2n$ sized subgraph that preserves distances within a multiplicative factor of 3 is known [8]. Braunschvig *et al.* [11] initiated the study of additive spanners. For β-additive spanner, Parter and Peleg [29] provided a $\Omega(n^{1+\epsilon(\beta)})$ size lower bound where $\epsilon(\beta) \in (0, 1)$. For single source undirected graphs they also constructed a 4-additive spanner of size $O(n^{4/3})$ that is resilient to single edge failure. For single vertex failure, constructions of additive spanners were given in [26, 7]. Very recently, for any fixed $k \geq 1$, construction of a sub-quadratic size 2-additive spanner resilient to k edge or vertex failures has been shown for unweighted undirected graphs [10]. In the same paper, authors also show that to achieve $O(n^{2-\epsilon})$ upper bound, one must allow $\Omega(\epsilon k)$ additive error.

Designing distance oracle is another important problem and also has been studied in edge failure model. The objective is to build a fault tolerant data structure that can answer queries about the distances in a given graph. For single edge failure the problem was first studied in [18]. Construction of $\widetilde{O}(n^2)$-space and $O(1)$-query time oracle is known for single edge failure due to [6]. In case of dual edge failures, near optimal $\widetilde{O}(n^2)$ size and $\widetilde{O}(1)$-query time oracle was given in [20]. The problem has also been studied under the restriction of bounded edge weights [22, 35]. For general k edge failures, Bilò *et al.* [9] gave a construction of $O(kn \log^2 n)$ size data structure that can report distance from a single source within multiplicative factor of $(2k + 1)$ in time $O(k^2 \log^2 n)$.

Another closely related problem is the replacement path problem where given a source and destination vertex and an edge, the objective is to find a path from source to destination avoiding that particular given edge. Though the problem was initially defined for single edge failure, later it was extended to multiple edge failures also. Readers may refer to [36, 31, 22, 35] for recent progresses on this problem.

Our technique

Before exhibiting the technique behind our result, we first state a simple observation. If we just store any shortest path tree rooted at s, then even after k weight increment that tree will preserve the distance from s to t (for any t) within k additive error. It is also necessary to include a shortest path tree inside a k-WTSS, otherwise we can never hope to get exact distances even when $k = 0$. Now since weight of any path can be increased by at most k, after including any shortest $s - t$ path in a k-WTSS it is not required to include another $s - t$ path that has weight more than or equal to $dist(s, t) + k$.

We argue that for the construction of k-WTSS it suffices to concentrate on any single vertex t and try to build a subgraph such that the distance between s and t is preserved under weight increment and we call such a subgraph a k-WTSS(t). This is because of the application of Locality Lemma (see Section 4), a variant of which also appears in [28, 29, 4]. Locality Lemma actually says slightly more, that if we can construct such a subgraph for any vertex t with an additional property that in-degree of t in the subgraph is bounded by some value c, then we can get a k-WTSS of size at most cn.

So from now on we can only talk about constructing k-WTSS(t). Let us take a toy example which provides a motivation behind our technique. Let the input graph be G and $dist(s, t) = d$. Suppose G is such that it can be decomposed into $k + 1$ disjoint subgraphs $G_0, \cdots, G_k$ where for $0 \leq i \leq k - 1$, G_i contains all the $s - t$ paths of weight $d + i$ present in G and any $s - t$ path in G_i has weight exactly $d + i$. In general such a decomposition may not exist. However, if it exists then it is not hard to get such a decomposition. Now given such a decomposition, we compute a k-FTRS(t) of G_0 and for $i \in [k - 1]$, a $(k - i - 1)$-FTRS(t) of G_i and then take the union of them. We claim that the obtained subgraph will be a k-WTSS(t). Say after weight increment, the (shortest) distance between s and t is $d + j$ for $1 \leq j < k$. Our assumption on j is justified because $j = 0, k$ cases are trivial as we have included k-FTRS(t). For a similar reason we can also assume that all the original shortest paths now have weight at least $d + j + 1$. Without loss of generality we further assume that no weight increment happens on the edges of the current shortest path. The justification of this assumption is provided in the full version [13]. Due to our assumption on decomposition of G, we know that the total increase in weight on the edges of G_j is bounded by $k - (j + 1)$ which also implies that at most $k - (j + 1)$ many edges of G_j are affected by weight increment. This is because our increment function is integer valued. Note that this is the place where integer valued increment plays a crucial role. However by our construction, we have included $(k - j - 1)$-FTRS(t) of G_j in our subgraph. Thus even if we remove those affected edges, since there is a path in G_j on which there is no weight increment, by the definition of $(k - j - 1)$-FTRS(t) there will be a surviving path included in our constructed subgraph. This proves the correctness. Also by the result of [4], in-degree of t of each $(k - i - 1)$-FTRS(t) of G_i is bounded by 2^{k-i-1} and hence total in-degree of t in the constructed k-WTSS(t) is bounded by 2^{k+1}. Hence we get a k-WTSS of size at most $2^{k+1}n$.

We have already mentioned that there may not exist the above type of decomposition for an arbitrary graph. In general, if we consider a subgraph by taking all the $s - t$ paths upto some specific weight, then that particular subgraph may also contain a $s - t$ path with larger weight. At this point the argument stated in the last paragraph fails completely. However, the nice thing is that if we just consider all the shortest paths and build a subgraph then it is true that there will not be any $s - t$ path of larger weight in that subgraph. Now if we use the construction of k-FTRS on this shortest path subgraph, then we can guarantee the preservation of distances as long as the distances do not change even after the weight increment. Though if the distance changes, we cannot say anything. This is the main challenge that we overcome in our algorithm. For that purpose we use the properties of the farthest min-cut of the shortest path subgraph.

Baswana *et al.* [4] used the concept of farthest min-cut to construct k-FTRS. In their work, they first computed a series of k farthest min-cuts by taking source sets in some nested fashion. Then they calculated a max-flow from the final source set and kept the incoming edges of t having non-zero flow. We further exploit their technique in this paper to get our algorithm. We consider the shortest path subgraph and then compute a series of farthest min-cuts similar to [4]. However as mentioned in the last paragraph, in this way we just get k-FTRS(t) of the shortest path subgraph. Now let us take the farthest min-cut considering s as source. Since it is a (s, t)-cut of the shortest path subgraph, removal of it destroys all the shortest $s - t$ paths present in the original graph. Now if we again compute the shortest path subgraph, we will get a subgraph containing only $s - t$ paths of weight $d + i$, for some $i > 0$. Then we can process this new subgraph as before to compute a sequence of k farthest min-cuts and remove the first one. We proceed in this way until we reach at a point that we are left with $s - t$ paths of weight at least $d + k$.

Now let us compare the situation with our previously described toy example. Removal of cut edges only helps us to generate some subgraph of each of G_i's. However computing k-FTRS(t) of just some subgraph of G_i's may not be sufficient to get k-WTSS(t). Thus for each G_i, we try to get a lot of subgraphs of it so that when we combine k-FTRS(t) of all of them, we get the same advantage that we got from computing $(k - i - 1)$-FTRS(t) of G_i in the toy example. One way of getting a lot of subgraphs of G_i is to try out removal of different cuts (not just the farthest one). Obviously we cannot try for all possible cuts, because there can be too many. Moreover, each time to reach at a subgraph of weight $d + i$ we may have to remove a series of $i - 1$ cuts. As a result we may end up with exponentially many choices on removal of cuts to get all possible subgraphs of G_i.

The good thing is that it suffices to use just a series of k farthest min-cuts computed before for the purpose of removal. A stronger claim is formally stated in Lemma 15. This will reduce the number of choices to only k^i for any fixed G_i. In our algorithm we establish a slightly better bound on the number of subgraphs of G_i needed to be considered to construct a k-WTSS(t). In the proof we use k-tuples to efficiently enumerate all of these subgraphs. Informally, a subgraph indexed by a specific k-tuple consists of only the $s - t$ paths survived after removal of a set of cut sets identified by the value of the coordinates of the k-tuple. Now after getting those subgraphs we apply a construction similar to that of k-FTRS(t) from [4] to get a bound on in-degree of t. We emphasize that actually we cannot directly apply algorithm of [4] in a black box fashion on each of the subgraphs of G_i that we consider, because in that case it will not give us the claimed bound.

In this paper we consider k-WTSS with respect to the weight increment on edges. Instead, it is also possible to take weights on the vertices and perform increment over them. However, one can directly apply our result by splitting each vertex v into two vertices v_i and v_o where all the incoming and outgoing edges of v are respectively directed into v_i and directed out of v_o, and then considering an edge (v_i, v_o) with the weight equal to that on v.

Organization of the paper

We discuss useful notations and some already known results about farthest min-cut in Section 2. Then in Section 3 we provide an algorithm to compute farthest min-cut of the shortest path subgraph and a few important properties about it. Next in Section 4, we reduce the problem of finding k-WTSS to that of finding k-WTSS(t) for some specific vertex t using Locality Lemma. Finally we prove Theorem 2 in Section 5. We also present several lower bound results in Section 6.

2 Preliminaries

Notations:

For any positive integer r, we denote the set $\{1, 2, \cdots, r\}$ by $[r]$. Throughout this paper we use $\mathbb{N}$ to indicate the set of natural numbers including zero. For any k-tuple σ and $i \in [k]$, we use the notation $\sigma(i)$ to denote the value of the i-th coordinate of σ. Given a directed graph $G = (V, E)$ on a set of vertices of size $|V| = n$ and a set of edges of size $|E| = m$ with a weight function w defined on the set of edges, a source vertex $s \in V$ and a destination vertex $t \in V$, we use the following notations throughout this paper.

- $V(G), E(G)$: the set of vertices and edges of G respectively.
- $w(P)$: weight of any path P.
- $dist_{G,w}(x, y)$: the shortest distance between any two vertices x and y in G when weight of each edge is defined by the weight function w.
- $G + (u, v)$: the graph obtained by adding an edge (u, v) to the graph G.
- $G \setminus F$: the graph obtained by removing the set of edges F from the graph G.
- $Out(A)$: the set of all vertices in $V \setminus A$ having an incoming edge from $A \subseteq V$.
- $In\text{-}Edge(A)$: the set of edges incoming to $A \subseteq V$.
- $P[x, y]$: the subpath of a path P from a vertex x to y.
- $P \circ Q$: the path formed by concatenating paths P and Q assuming the fact that last vertex of P is same as first vertex of Q.
- $E(f)$: support of the flow f, i.e., the set of edges e such that $f(e) \neq 0$.
- $MaxFlow(G, S, t)$: any maximum valued flow in G from a source set S to t.
- G_{short} : the shortest path subgraph of G, i.e., union of all shortest $s - t$ paths in G.
- $ShortMaxFlow(G, S, t)$: any maximum valued flow returned by $MaxFlow(G_{short}, S, t)$.

The following definition introduces the notion of k-WTSS with respect to a fixed vertex t.

▶ **Definition 4** (k-WTSS(t))**.** Given a graph G with a weight function w, a source vertex $s \in V(G)$, another vertex $t \in V(G)$ and an integer $k \geq 1$, a subgraph $H_t = (V(G), E')$ where $E' \subseteq E(G)$ is said to be a *k-WTSS(t)* of G if for any weight increment function $I : E(G) \to \mathbb{N}$ such that $\sum_{e \in E(G)} I(e) \leq k$, the following holds: for the weight function defined by $w'(e) = w(e) + I(e)$ for all $e \in E(G)$, $dist_{G,w'}(s, t) = dist_{H_t,w'}(s, t)$.

The restriction on the range of the increment function to $\mathbb{N}$ is justified because only in that case we can hope for a sparse k-WTSS(t) (see Section 6). However one can easily extend the above definition for increment function $I : E(G) \to \mathbb{R}$. Following is an alternative definition of k-WTSS in terms of k-WTSS(t).

▶ **Definition 5.** A subgraph H is a *k-WTSS* of G iff it is a k-WTSS(t) for all $t \in V(G)$.

2.1 Max-flow and farthest min-cut

The algorithm described in this paper heavily exploits the connection between min-cut, max-flow and the number of edge disjoint paths present in a graph. Let us start with the following well known fact of Graph Theory.

▶ **Theorem 6.** *In any graph with unit capacity on edges, there is a flow of value r from a source set S to a destination vertex t if and only if there exist r edge disjoint paths that originate from the set S and terminate at t.*

For the sake of clarity, we emphasize that though we talk about weighted graphs, throughout this paper we will use capacity functions on edges that take values only from $\{0, 1\}$.

▶ **Definition 7** ((S, t)-min-cut)**.** In any graph G an (S, t)*-cut* is a set of edges $C \subseteq E(G)$ such that every path from any vertex $s \in S$ to t must pass through some edge in C. An (S, t)-cut is called (S, t)*-min-cut* if it has the smallest size among all other (S, t)-cuts.

Any (S, t)-cut C partitions the vertex set $V(G)$ into two subsets $A(C)$ and $B(C)$ where $A(C)$ is the set of all the vertices reachable from S in $G \setminus C$ and $B(C) = V(G) \setminus A(C)$. Note that $S \subseteq A(C)$ and $t \in B(C)$. From now on, we assume this pair of vertex sets $(A(C), B(C))$ to be output of a function $Partition(G, C)$. For our purpose we do not just consider any (S, t)-min-cut, instead we consider the farthest one.

▶ **Definition 8** (Farthest Min Cut [21])**.** Let S be a source set and t be a destination vertex in any graph G and suppose for any (S, t)-min-cut C, $(A(C), B(C)) = Partition(G, C)$. Any (S, t)-min-cut C_{far} is called *farthest min-cut*, denoted by $FMC(G, S, t)$, if for any other (S, t)-min-cut C, it holds that $A(C) \subsetneq A(C_{far})$.

The following lemma given by Ford and Fulkerson [21] establishes the uniqueness of farthest min-cut and also provides an algorithm to compute it.

▶ **Lemma 9** ([21])**.** *Suppose f be a max-flow in G from any source set S to t and G_f be the corresponding residual graph. If B is the set of vertices from which there is a path to t in G_f and $A = V(G) \setminus B$, then the set C of edges that start at A and terminate at B is the unique farthest (S, t)-min-cut.*

3 Farthest Min-cut of Shortest Path Subgraph

3.1 Computing farthest min-cut of shortest path subgraph

In this section we give an algorithm to find the farthest min-cut of the shortest path subgraph of a given graph. We are given a weighted directed graph G with two vertices s and t. The weight of each edge of G is defined by a weight function $w : E(G) \to \mathbb{R}$. Let $dist_{G,w}(s, t) = d$. We denote the set of all $s - t$ paths of weight d under the weight function w by $\mathcal{P}_d$ and the corresponding underlying subgraph (just the union of all the paths in $\mathcal{P}_d$) of G by G_{short}. More specifically, $V(G_{short}) = V(G)$ and $E(G_{short}) = \{e \mid e \in P \text{ for some } P \in \mathcal{P}_d\}$. For any graph G and two vertices s and t, for any source set $S \subseteq V(G_{short})$, *farthest min-cut of shortest path subgraph*, denoted as $FSMC(G, s, S, t)$ is defined by $FMC(G_{short}, S, t)$.

For any (S, t)-cut C of G_{short} we define the partition function by $ShortPartition(G, C) = Partition(G_{short}, C)$. It is easy to design (see the full version [13]) a simple $O(mn)$ time procedure which given G, s and t, generates the subgraph G_{short}. Then we can simply apply well known Ford-Fulkerson algorithm [21] on the subgraph G_{short} to find the $ShortMaxFlow(G, S, t)$ and $FSMC(G, s, S, t)$. The correctness of $FSMC(G, s, S, t)$ follows from applying Lemma 9 on the subgraph G_{short}.

3.2 Disjoint shortest path lemma

Let us choose any $r \in \mathbb{N}$ and then consider the following: Set $S_1 = \{s\}$ and for $i \in [r]$, define $C_i = FSMC(G, s, S_i, t)$, $(A_i, B_i) = ShortPartition(G, C_i)$ and $S_{i+1} = (A_i \cup Out(A_i)) \setminus \{t\}$. Let $E' \subseteq E(G)$ such that $E' = \{(u_1, v_1), \cdots, (u_r, v_r)\}$, where $(u_i, v_i) \in C_i$.

Now let us introduce an auxiliary graph $G' = G + (s, v_1) + \cdots + (s, v_r)$ and set $w(s, v_i) = dist_{G,w}(s, v_i)$ for $i \in [r]$. Suppose f is a max-flow from S_{r+1} to t in the shortest path subgraph of G and $\mathcal{E}(t)$ be the set of incoming edges of t having nonzero flow value assigned by f. Now consider a new graph $G^* = (G' \setminus In\text{-}Edge(t)) + \mathcal{E}(t)$.

▶ **Lemma 10.** *There will be at least $r + 1$ disjoint paths in G^* each of weight equal to $dist_{G,w}(s, t)$.*

Note that a similar claim was shown in [4]. However, our claim is slightly more general because we consider an edge set E' where the edges belong to E' may not lie on a single $s - t$ path in G and also we comment on the weight of the disjoint paths. Both of these requirements are crucial for the proof in Section 5. Fortunately, the proof in [4] does not rely on the fact that the edges (u_i, v_i)'s are part of a single $s - t$ path.

4 Construction of k-WTSS and Locality Lemma

Let us first recall the problem. We are given a graph G along with a weight function $w : E(G) \to \mathbb{Z}$ and a source vertex s. Now suppose for every $e \in E(G)$, $w(e)$ is increased by some arbitrary weight increment function $I : E(G) \to \mathbb{N}$ such that total increase in weight is bounded by k, i.e., $\sum_{e \in E(G)} I(e) \leq k$ and we denote the new weight function (after increase in weight) by w' where $w'(e) = w(e) + I(e)$. The problem is to find a subgraph H such that for any vertex $t \in V(G)$ in H there always exists an $s - t$ path of weight $dist_{G,w'}(s, t)$. We call this subgraph H a k-WTSS. Now if we just want the requirement of existence of a path in H to be true for a fixed vertex t instead of all vertices, then we call such a subgraph k-WTSS(t). In this section we reduce the problem of finding k-WTSS to the problem of finding k-WTSS(t) for any fixed vertex $t \in V(G)$. The following lemma, a variant of which also appears in [4], serves our purpose.

▶ **Lemma 11** (Locality Lemma). *Let there be an algorithm $\mathcal{A}$ that given a graph G and a vertex $t \in V(G)$, generates a subgraph H_t of G such that:*
- *H_t is a k-WTSS(t); and*
- *in-degree of t in H_t is bounded by a constant c_k.*

Then one can generate a k-WTSS of G such that it has only $c_k \cdot n$ edges.

5 Construction of k-WTSS(t)

In this section we provide an algorithm to compute a k-WTSS(t) for any fixed vertex $t \in V(G)$ where source vertex is s. Without loss of generality let us first assume the following.

▶ **Assumption 12.** *The out degree of source vertex s is 1 and the out degree of all other vertices is bounded by 2.*

For any graph G if $|Out(s)| > 1$ then to satisfy our previous assumption, we can simply add a new vertex s_0 and add an edge (s_0, s) and set $w(s_0, s) = 0$. Then make this new vertex s_0 as our new source. For the justification on the bound on out degree of other vertices, we refer the readers to the full version [13]. Note that the reduction process blows up the number of

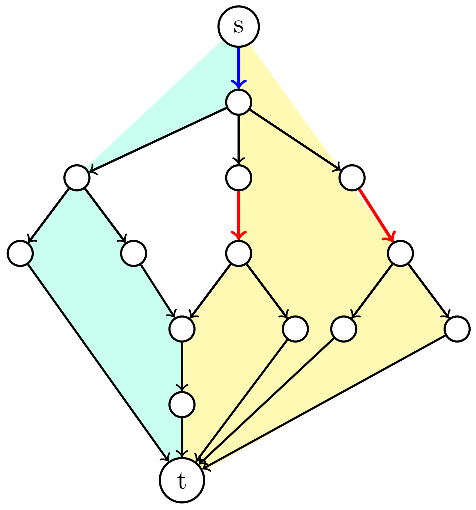

Figure 1 Region shaded with green color represents G_σ for $\sigma = (1, -1, \cdots, -1)$ whereas yellow colored region is the shortest path subgraph of G. The edges of $C_{(-1,\cdots,-1),1}$ and $C_{(-1,\cdots,-1),2}$ are colored with blue and red respectively. G_σ is obtained by removing red colored edges.

vertices from n to $O(m)$. However, since we bound the in-degree of t in k-WTSS(t) for any $t \in V(G)$, by a value independent of the number of vertices in the graph, our result remains unaffected.

5.1 Description of the algorithm

Before describing the algorithm let us introduce some notations that we will use later heavily. Consider any k-tuple $\sigma \in \{-1, 0, 1, \cdots, k\}^k$ such that if $\sigma(i) = -1$ (where $\sigma(i)$ denotes the i-th coordinate of σ) then for all $i' > i$, $\sigma(i') = -1$ and if $\sigma(i) \neq -1$ then for all $i' < i$, $\sigma(i') \neq -1$. We use these k-tuples to efficiently enumerate all the subgraphs of G for which we want to calculate farthest min-cuts. Informally, a subgraph indexed by σ consists of only the $s - t$ paths survived after removal of a set of cut sets identified by $\sigma(i)$'s. For any such $\sigma \in \{-1, 0, 1, \cdots, k\}^k$ and $r \in [k]$, we recursively define the subgraph G_σ, set of source vertices $S_{\sigma,r}$ and edge set $C_{\sigma,r}$ as follows: if $\sigma = (-1, -1, \cdots, -1)$, G_σ is the union of all $s-t$ paths in G, starting with $S_{\sigma,1} = \{s\}$, for any $r \in [k]$ define $C_{\sigma,r} = FSMC(G_\sigma, s, S_{\sigma,r}, t)$, $S_{\sigma,r+1} = (A \cup Out(A)) \backslash \{t\}$ where $(A, B) = ShortPartition(G_\sigma, C_{\sigma,r})$. For $\sigma \neq (-1, \cdots, -1)$, G_σ is the union of all $s - t$ paths in $G_{\sigma'} \setminus C_{\sigma',\sigma(i)+1}$, where $i = \min\{i' \mid \sigma(i') = -1\}$ and

$$\sigma'(i') = \begin{cases} \sigma(i') & \text{if } i' < i - 1 \\ -1 & \text{otherwise} \end{cases}$$

Now starting with $S_{\sigma,1} = \{s\}$, if there exists a $s - t$ path of weight $d + i - 1$ then for any $r \in [k]$ define $C_{\sigma,r} = FSMC(G_\sigma, s, S_{\sigma,r}, t)$, $S_{\sigma,r+1} = (A \cup Out(A)) \setminus \{t\}$ where $(A, B) = ShortPartition(G_\sigma, C_{\sigma,r})$; else set $C_{\sigma,r} = \phi$. We refer the reader to Figure 1 for the better understanding about the graph G_σ.

We are given a weighted directed graph G with a weight function w and a source vertex s and a destination vertex t. The weight of each edge of G is defined by the weight function $w : E(G) \to \mathbb{Z}$. Overall our algorithm performs the following tasks: For different values of $\sigma \in \{-1, 0, \cdots, k\}^k$ it computes the sets $C_{\sigma,i}$ and $S_{\sigma,i}$ for $i \in [k]$. Then for each such σ, it computes max-flow in the shortest path subgraph of G_σ by considering $S_{\sigma,k}$ as source and add the edges incident on t with non-zero flow to a set $\mathcal{E}(t)$. At the end, our algorithm returns the subgraph $H_t = (G \setminus \mathit{In\text{-}Edge}(t)) + \mathcal{E}(t)$.

Our algorithm performs the above tasks in the recursive fashion. Starting with $\sigma = (-1, \cdots, -1)$, it first considers the shortest path subgraph of $G_\sigma = G$ and performs k iterations on it. At each iteration it computes the farthest min-cut $C_{\sigma,i}$ by considering $S_{\sigma,i}$ as source and t as sink starting with $S_{\sigma,1} = \{s\}$. Then it updates the graph by removing the edges present in $C_{\sigma,i}$ and passes this new graph in the next recursive call. Before the recursive call it also updates the σ by incrementing the value of $\sigma(j)$ by one and passes the updated value of σ to the recursive call. Here j is a parameter which denotes that the smallest coordinate of σ that has value -1. Initially j was set to 1 and before the next recursive call we increment its value by one. At the end of each iteration our algorithm updates the source set to $S_{\sigma,i+1}$ by including end points of all the edges present in the cut $C_{\sigma,i}$ in the set $S_{\sigma,i}$. At the end of k iterations, the algorithm computes max-flow in the shortest path subgraph of G_σ by considering $S_{\sigma,k}$ as source and add the edges incident on t with non-zero flow to a set $\mathcal{E}(t)$.

5.2 Correctness proof

Let us start with the following simple observation.

▶ **Observation 13.** *For any $\sigma \in \{-1, 0, \cdots, k\}^k$, any $s-t$ path in G_σ must have weight at least $d + i - 1$ where $i = \min\{i' \mid \sigma(i') = -1\}$.*

Note that the above observation is true only because we consider the range of our weight function w to be $\mathbb{Z}$. Otherwise above observation will trivially be false.

Now let us consider any increment function $I : E(G) \to \mathbb{N}$ such that $\sum_{e \in E(G)} I(e) \le k$ and then denote the set of edges with non-zero value of the function I by F, i.e., $F = \{e \in E(G) | I(e) > 0\}$. So clearly $|F| \le k$. Now suppose $dist_{G,w'}(s,t) = d' = d + j$ for some $0 \le j \le k$ where $w'(e) = w(e) + I(e)$. Thus we need to show that there also exists an $s-t$ path of weight d' in the subgraph H_t under the new weight function w'.

Suppose P be an $s-t$ path in G such that $w'(P) = d' = d + j$. For simplicity let us make the following assumption.

▶ **Assumption 14.** *For all $e \in P$, $I(e) = 0$, i.e., $w'(P) = w(P)$.*

▶ **Lemma 15.** *One of the following three cases must be satisfied.*
1. *There exists a σ such that P belongs to the subgraph G_σ where $\sigma(j) = -1$ and for some $r \in [k]$, the last edge of P belongs to the edge set $C_{\sigma,r}$.*
2. *There exists a σ such that P belongs to the subgraph G_σ where $\sigma(j+1) = -1$, $\sigma(j) \ne -1$ and there is no $i \in [j-1]$ such that $\sigma(i) \ge k - j + i - 1$.*
3. *There exists a σ such that P belongs to the subgraph G_σ where if $i = \min\{i' \mid \sigma(i') = -1\}$ then $i \le j$ and for all $i' < i$, $\sigma(i') < k - j + i' - 1$ and P passes through all the cut sets $C_{\sigma,1}, \cdots, C_{\sigma,k-j+i-1}$.*

Now let us call the path P is of type-1, type-2 and type-3 respectively depending on which of the above three cases it satisfies.

Type-1. This case is the simplest among the three.

▶ **Lemma 16.** *If P is a type-1 path then P is contained in the subgraph H_t.*

Proof. Suppose (v,t) is the last edge of the path P. Now since $(v,t) \in C_{\sigma,r}$ for some σ and r, $(v,t) \in \mathcal{E}(t)$. Thus by the construction of the subgraph H_t, the edge (v,t) belongs to H_t. Also by the construction of the subgraph H_t, for all the vertices $u \neq t$, *In-Edge*(u) belongs to H_t. Hence P must lie completely inside H_t. ◀

Type-2. By Observation 13 any path that belongs to the subgraph G_σ where $\sigma(j) \neq -1$ and $\sigma(j+1) = -1$, must have weight atleast $d+j$ under the weight function w. Now, since by Assumption 14 $w(P) = d+j$, P must pass through an edge $(u_r, v_r) \in C_{\sigma,r}$ for all $r \in [k]$. Consider an auxiliary graph $G'_\sigma = G_\sigma + (s,v_1) + \cdots + (s,v_k)$ and extend the weight function w as $w(s,v_r) = w(P[s,v_r])$. Then define another graph $G^*_\sigma = (G'_\sigma \setminus \textit{In-Edge}(t)) + \mathcal{E}(t)$. By Lemma 10 we claim the following.

▶ **Corollary 17.** *There will be $k+1$ edge disjoint paths in G^*_σ each of weight $w(P)$ under weight function w.*

Now we use the above corollary to conclude the following.

▶ **Lemma 18.** *If P is a type-2 path then there exists an $s-t$ path of weight d' in the subgraph H_t under the new weight function w'.*

Proof. By Corollary 17, we get $k+1$ edge disjoint paths $P_1, \cdots, P_{k+1}$ each of weight $w(P) = d+j$. Since $|F| \leq k$ where $F = \{e \in E(G) | I(e) > 0\}$, at least one of the $k+1$ edge disjoint paths, say P_1, must survive in $G^*_\sigma \setminus F$. If P_1 also belongs to the subgraph H_t then we are done. Otherwise P_1 must take some of the (s,v_r)'s as the first edge and the remaining portion $P_1[v_r,t]$ lies inside H_t. Now consider the following path $R = P[s,v_r] \circ P_1[v_r,t]$. By the construction of G'_σ, $w'(R) = w'(P_1) = w(P)$ and this completes the proof. ◀

Type-3. Suppose P is a type-3 path and thus belongs to G_σ for some σ where if $i = \min\{i' \mid \sigma(i') = -1\}$ then $i \leq j$ and for any $i' \leq i$, $\sigma(i') < k-j+i'-1$ and P passes through all the cut sets $C_{\sigma,1}, \cdots, C_{\sigma,k-j+i-1}$. P passes through an edge $(u_r,v_r) \in C_{\sigma,r}$ for all $r \in [k-j+i-1]$. For the ease of representation let us define $v_0 = s$. Now if there exists a positive integer $r \in [k-j+i-1]$ such that $w(P[v_{r-1},u_r]) > dist_{G_\sigma,w}(v_{r-1},u_r)$, replace the portions of path $P[v_{r-1},u_r]$ by the $v_{r-1} - u_r$ path of weight $dist_{G_\sigma,w}(v_{r-1},u_r)$. We do this until there is no such r and after that we call this new path as P'.

Now consider an auxiliary graph $G'_\sigma = G_\sigma + (s,v_1) + \cdots + (s,v_{k-j+i-1})$ and extend the weight function w as $w(s,v_r) = w(P[s,v_r])$. Next define another graph $G^*_\sigma = (G'_\sigma \setminus$ *In-Edge*$(t)) + \mathcal{E}(t)$. Now we use a similar but slightly more intricate argument than that used in case of Type-2 paths to conclude the following.

▶ **Lemma 19.** *If P is a type-3 path then there exists an $s-t$ path of weight $w(P) = d'$ in the subgraph H_t under the new weight function w'.*

5.3 Bound on the size of $\mathcal{E}(t)$

Before establishing the upper bound on the size of the set of edges $\mathcal{E}(t)$, let us define $C_{\sigma,k+1} = FSMC(G_\sigma, s, S_{\sigma,k+1}, t)$ for any $\sigma \in \{-1, 0, \cdots, k\}^k$.

Now since $FSMC(G_\sigma, s, S_{\sigma,i+1}, t) = FMC(G^{short}_\sigma, S_{\sigma,i+1}, t)$ for any $i \in [k]$ where G^{short}_σ is the shortest path subgraph of G_σ, we can restate Lemma 6.6 from [4] as follows.

▶ **Lemma 20.** *For any $i \in [k]$, $|C_{\sigma,i+1}| \leq 2 \cdot |C_{\sigma,i}|$.*

Reader may note that the proof of the above lemma in [4] crucially relies on Assumption 12.

▶ **Lemma 21.** $|\mathcal{E}(t)| \leq e(k-1)!2^k$.

Proof. In our algorithm for each $\sigma \in \{-1, 0, \cdots, k\}^k$ we compute the cut sets $C_{\sigma,1}, \cdots, C_{\sigma,k}$ and add $|C_{\sigma,k+1}|$ many edges in the set $\mathcal{E}(t)$ only if for all $i' < i$, $\sigma(i') < k - i + i' - 1$ where $i = \min\{j \mid \sigma(j) = -1\}$; otherwise we do not compute anything. So the total number of σ for which we add edges in $\mathcal{E}(t)$ is bounded by

$$1+(k-1)+(k-1)(k-2)+\cdots+(k-1)! = (k-1)![1/0!+1/1!+\cdots+1/(k-1)!] \leq e\cdot(k-1)!.$$

Now by applying Lemma 20, we get that for each such σ, $|C_{\sigma,k+1}| \leq 2^k$ (since $|C_{\sigma,1}| = 1$) and this proves the claimed bound. ◀

Complexity analysis. Here we just mention that since by Lemma 11 finding k-WTSS requires n rounds where in each round we find k-WTSS(v) for some $v \in V(G)$, computing k-WTSS takes total $O(k^k m^2 n)$ time.

6 Lower Bound Results for the Size of k-WTSS

Theorem 3 already provides a lower bound on the size of k-WTSS with the restriction same as that considered in our k-WTSS construction. In this section we show that the size of k-WTSS of a graph can be of size at least $\Omega(n^2)$ even for $k = 1$ if we allow either the weight function or the increment function to be rational valued. We formally state the lower bounds in the following two theorems, proofs of which can be found in the full version [13].

▶ **Theorem 22.** *If weight of an edge can be any rational value, then for every $n \in \mathbb{N}$, there exists a directed graph with n vertices whose 1-WTSS must contain $c \cdot n^2$ many edges for some constant $c > 0$.*

▶ **Theorem 23.** *If it is allowed to increase the weight of the edges by any rational value, then for every $n \in \mathbb{N}$, there exists a directed graph with n vertices whose 1-WTSS must contain $c \cdot n^2$ many edges for some constant $c > 0$.*

▶ Remark. We emphasize that all the lower bound results in the above section hold for undirected graphs also. Moreover, exactly the same graphs without any direction will serve the purpose.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 351–361, 2016.

2 Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 568–576, 2017.

3 Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single source fault tolerant shortest path. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1901–1915, 2018.

4 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 509–518, 2016.

5 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (alpha, beta)-spanners and purely additive spanners. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005*, pages 672–681, 2005.

6 Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 101–110, 2009.

7 Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Proceedings*, pages 167–178, 2015.

8 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-tolerant approximate shortest-path trees. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Proceedings*, pages 137–148, 2014.

9 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Multiple-edge-fault-tolerant approximate shortest-path trees. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*, pages 18:1–18:14, 2016.

10 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, pages 73:1–73:14, 2017.

11 Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault tolerant additive and (μ, α)-spanners. *Theor. Comput. Sci.*, 580:94–100, 2015.

12 Luciana S. Buriol, Mauricio G. C. Resende, and Mikkel Thorup. Speeding up dynamic shortest-path algorithms. *INFORMS Journal on Computing*, 20(2):191–204, 2008.

13 Diptarka Chakraborty and Debarati Das. Near optimal sized weight tolerant subgraph for single source shortest path. *CoRR*, abs/1707.04867, 2017.

14 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault-tolerant spanners for general graphs. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 435–444, 2009.

15 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, pages 130:1–130:13, 2016.

16 Artur Czumaj and Hairong Zhao. Fault-tolerant geometric spanners. *Discrete & Computational Geometry*, 32(2):207–230, 2004.

17 Camil Demetrescu and Giuseppe F. Italiano. Fully dynamic all pairs shortest paths with real edge weights. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 260–267, 2001.

18 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

19 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011*, pages 169–178, 2011.

20 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 506–515, 2009.

21 D. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 2010.

22 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 748–757, 2012.

23 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, pages 127:1–127:15, 2017.

24 Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979.

25 Tamás Lukovszki. New results of fault tolerant geometric spanners. In *Algorithms and Data Structures, 6th International Workshop, WADS '99, Proceedings*, pages 193–204, 1999.

26 Merav Parter. Vertex fault tolerant additive spanners. In *Distributed Computing - 28th International Symposium, DISC 2014, Proceedings*, pages 167–181, 2014.

27 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*, pages 481–490, 2015.

28 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Proceedings*, pages 779–790, 2013.

29 Merav Parter and David Peleg. Fault tolerant approximate BFS structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1073–1092, 2014.

30 G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest path problem. *Journal of Algorithms*, 21:267–305, 1996.

31 Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Trans. Algorithms*, 8(4):33:1–33:11, 2012.

32 Hans Rohnert. A dynamization of the all pairs least cost path problem. In *STACS 85, 2nd Symposium of Theoretical Aspects of Computer Science, Proceedings*, pages 279–286, 1985.

33 Mohammadreza Saeedmanesh and Nikolas Geroliminis. Dynamic clustering and propagation of congestion in heterogeneously congested urban traffic networks. *Transportation Research Part B: Methodological*, 105(Supplement C):193–211, 2017.

34 Maxim Teslenko and Elena Dubrova. An efficient algorithm for finding double-vertex dominators in circuit graphs. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005)*, pages 406–411, 2005.

35 Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Trans. Algorithms*, 9(2):14:1–14:13, 2013.

36 Virginia Vassilevska Williams. Faster replacement paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 1337–1346, 2011.

37 David P. Woodruff. Additive spanners in nearly quadratic time. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010*, pages 463–474, 2010.