Network Flow Spanners*

Feodor F. Dragan, Chenyu Yan
Department of Computer Science, Kent State University, Kent, Ohio 44242

In this article, motivated by applications of ordinary (distance) spanners in communication networks and to address such issues as bandwidth constraints on network links, link failures, network survivability, etc., we introduce a new notion of flow spanner, where one seeks a spanning subgraph $H = (V, E')$ of a graph $G = (V, E)$ which provides a "good" approximation of the source-sink flows in G. We formulate several variants of this problem and investigate their complexities. Special attention is given to the version where H is required to be a tree. © 2009 Wiley Periodicals, Inc. NETWORKS, Vol. 00(00), 000–000 2009

Keywords: network design; maximum flow preservation; spanners; spanning trees; approximation algorithms; NP-completeness

1. INTRODUCTION

Given a graph $G = (V, E)$, a spanning subgraph $H = (V, E')$ of G is called a spanner if H provides a "good" approximation of the distances in G. More formally, for $t \geq 1$, H is called a t–spanner of G [9, 29, 30] if $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$, where $d_G(u, v)$ is the distance in G between u and v. Sparse spanners (where $|E'| = O(|V|)$) have applications in various areas; especially, in distributed systems and communication networks. In [30], close relationships were established between the quality of spanners (in terms of stretch factor t and the number of spanner edges $|E'|$), and the time and communication complexities of any synchronizer for the network based on this spanner. Sparse spanners are also very useful in message routing in communication networks; to maintain succinct routing tables, efficient routing schemes can use only the edges of a sparse spanner [31]. It is known that the problem of determining, for a given graph G and two integers $t, m \geq 1$, whether G has a t-spanner with m or fewer edges, is NP-complete (see [29]).

The sparsest spanners are tree spanners. Tree spanners occur in biology [2], and as it was shown in [28], they can be used as models for broadcast operations. Tree t-spanners were considered in [6]. It was shown that, for a given graph G, the problem to decide whether G has a spanning tree T such that $d_T(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$ is NP–complete for any fixed $t \geq 4$ and is linearly solvable for $t = 1, 2$. For more information on spanners consult [1, 3–7, 9, 12, 13, 15, 26, 28–31, 33, 34].

In this article, motivated by applications of spanners in communication networks and to address such issues as bandwidth constraints on network links, link failures, network survivability, etc., we introduce a new notion of flow spanner, where one seeks a spanning subgraph $H = (V, E')$ of a graph G which provides a "good" approximation of the source-sink flows in G. We formulate several variants of this problem and investigate their complexities. In this preliminary investigation, special attention is given to the version where H is required to be a tree.

2. PROBLEM FORMULATIONS AND RESULTS

A network is a 4-tuple $N = (V, E, c, p)$ where $G = (V, E)$ is a connected, finite, and simple graph, c(e) are nonnegative edge *capacities*, and $p(e)$ are nonnegative edge *prices*. We assume that graph G is undirected in this article, although similar notions can be defined for directed graphs as well. In this case, $c(e)$ indicates the maximum amount of flow edge $e = (v, u)$ can carry (in either v to u direction or in u to v direction), $p(e)$ is the cost that the edge will incur if it carries a non-zero flow. Given a source s and a sink t in G, an (s, t)-*flow* is a function f defined over the edges that satisfies capacity constraints, for every edge, and conservation constraints, for every vertex, except the source and the sink. The net flow that enters the sink t is called the (s, t)-*flow*. Denote by $F_G(s, t)$ the *maximum* (s, t)-*flow* in G. Note that, since G is undirected, $f(v, u) = -f(u, v)$ for any edge $e = (v, u) \in E$ and $F_G(x, y) = F_G(y, x)$ for any two vertices (source and sink) x and y (by reversing the flow on each edge).

Let $H = (V, E')$ be a subgraph of G, where $E' \subseteq E$. For any two vertices $u, v \in V(G)$, define flow–stretch$(u, v) = \frac{F_G(u,v)}{F_H(u,v)}$ to be the flow–stretch *factor* between u and v. Define the flow–stretch *factor* of H as

$$fs_H = \max\{flow-stretch(u, v) | \forall u, v \in V(G)\}.$$

When the context is clear, the subscript H will be omitted.

Received October 2006; accepted July 2009
Correspondence to: F.F. Dragan; e-mail: dragan@cs.kent.edu
*Results of this article were presented at the LATIN'06 conference [10].
DOI 10.1002/net.20357
Published online in Wiley InterScience (www.interscience.wiley.com).

Similarly, define the *average* flow–stretch *factor* of the subgraph H as follows

$$afs_H = \frac{2}{n(n-1)} \sum_{u,v \in V} \frac{F_G(u,v)}{F_H(u,v)}.$$

The general problem, we are interested in, is to find a light flow–spanner H of G, that is a spanning subgraph H such that fs_H (or afs_H) is as small as possible and at the same time the total cost of the spanner, namely

$$\mathcal{P}(H) = \sum_{e \in E'} p(e),$$

is as low as possible. The following is the decision version of this problem.

Problem: Light Flow–Spanner

Instance: An undirected graph $G = (V, E)$, nonnegative edge capacities $c(e)$, non-negative edge costs $p(e)$, $e \in E(G)$, and two positive numbers t and B.

Output: A light flow–spanner $H = (V, E')$ of G with flow–stretch factor $fs_H \leq t$ and total cost $\mathcal{P}(H) \leq B$, or "there is no such spanner."

We distinguish also few special variants of this problem.

Problem: Sparse Flow–Spanner

Instance: An undirected graph $G = (V, E)$, non-negative edge capacities $c(e)$, unit edge costs $p(e) = 1$, $e \in E(G)$, and two positive numbers t and B.

Output: A sparse flow–spanner $H = (V, E')$ of G with flow–stretch factor $fs_H \leq t$ and $\mathcal{P}(H) = |E'| \leq B$, or "there is no such spanner."

Problem: Sparse Edge-Connectivity–Spanner

Instance: An undirected graph $G = (V, E)$, unit edge capacities $c(e) = 1$, unit edge costs $p(e) = 1$, $e \in E(G)$, and two positive numbers t and B.

Output: A sparse flow–spanner $H = (V, E')$ of G with flow–stretch factor $fs_H \leq t$ and $\mathcal{P}(H) = |E'| \leq B$, or "there is no such spanner."

Note that here, the maximum (s, t)-flow in H is actually the maximum number of edge-disjoint (s, t)-paths in H, i.e., the edge-connectivity of s and t in H. Thus, this problem is named the Sparse Edge-Connectivity–Spanner problem. Spanning subgraph H provides a "good" approximation of the vertex-to-vertex edge-connectivities in G. The following is the version of this Edge-Connectivity Spanner problem with arbitrary costs on edges.

Problem: Light Edge-Connectivity–Spanner

Instance: An undirected graph $G = (V, E)$, unit edge capacities $c(e) = 1$, arbitrary non-negative edge costs $p(e)$, $e \in E(G)$, and two positive numbers t and B.

Output: A light flow–spanner $H = (V, E')$ of G with flow–stretch factor $fs_H \leq t$ and total cost $\mathcal{P}(H) \leq B$, or "there is no such spanner."

In Section 4, using a reduction from the 3-dimensional matching problem, we show that the Sparse Edge-Connectivity–Spanner problem is NP-complete, implying that all other three problems, defined above, are NP-complete as well.

Replacing in all four formulations "$fs_H \leq t$" with "$afs_H \leq t$", we obtain four more variations of the problem: Light Average Flow–Spanner, Sparse Average Flow–Spanner, Sparse Average Edge-Connectivity–Spanner, and Light Average Edge-Connectivity–Spanner, respectively. These four problems are topics of our current investigations.

In Section 5, we investigate two simpler variants of the problem: Tree Flow–Spanner and Light Tree Flow–Spanner problems.

Problem: Tree Flow–Spanner

Instance: An undirected graph $G = (V, E)$, non-negative edge capacities $c(e)$, $e \in E(G)$, and a positive number t.

Output: A tree t-flow–spanner $T = (V, E')$ of G, that is a spanning tree T of G with flow–stretch factor $fs_T \leq t$, or "there is no such tree spanner".

Problem: Light Tree Flow–Spanner

Instance: An undirected graph $G = (V, E)$, non-negative edge capacities $c(e)$, non-negative edge costs $p(e)$, $e \in E(G)$, and two positive numbers t and B.

Output: A light tree t-flow–spanner $T = (V, E')$ of G, that is a spanning tree T of G with flow–stretch factor $fs_T \leq t$ and total cost $\mathcal{P}(T) \leq B$, or "there is no such tree spanner."

In a similar way one can define also the Tree Average Flow–Spanner and Light Tree Average Flow–Spanner problems. Notice that our tree t-flow-spanners are different from the well-known *Gomory-Hu* trees [21]. A Gomory-Hu tree gives a nice structure for representing in a compact way all *s-t* maximum flows of an undirected graph, but it is not necessarily a spanning tree of the graph.

We show that the Tree Flow–Spanner problem has easy polynomial time solution while the Light Tree Flow–Spanner problem is NP-complete. In Section 6, we propose some approximation algorithms for the Light Tree Flow–Spanner problem.

3. RELATED WORK

In [18], a network design problem, called *smallest k-edge connected spanning subgraph problem* (smallest k-ECSS problem) is considered, which is close to our Sparse Edge-Connectivity–Spanner problem. In that problem, given a graph G along with an integer k, one seeks a spanning subgraph H of G that is k-edge-connected and contains the fewest possible number of edges. The problem is known to be MAX SNP-hard [16], and the authors of [18] give a polynomial time algorithm with approximation ratio $1 + 2/k$ (see also [8] for an earlier approximation result). It is interesting to note that a sparse k-edge-connected spanning subgraph (with $O(k|V|)$ edges) of a k-edge-connected graph can be found in linear time [27]. In our Sparse Edge-Connectivity–Spanner problem, instead of trying to guarantee the k-edge-connectedness in H for all vertex pairs, we try to closely approximate by H the original (in G) levels of edge-connectivities.

Paper [20] deals with the *survivable network design problem (SNDP)* which can be considered as a generalization of our Light Edge-Connectivity–Spanner problem. In SNDP, we are given an undirected graph $G = (V, E)$, a non-negative cost $p(e)$ for every edge $e \in E$ and a non-negative connectivity requirement r_{ij} for every (unordered) pair of vertices i, j. One needs to find a minimum-cost subgraph in which each pair of vertices i, j is joined by at least r_{ij} edge-disjoint paths. The problem is NP-complete since the Steiner Tree Problem is a special case, and [17, 19, 20, 23, 24, 35] give different approximate solutions to the problem. The best approximation algorithm known is a 2-approximation algorithm due to Jain [23]. This algorithm improved upon a primal-dual $2\mathcal{H}(k)$-approximation algorithm for SNDP of Goemans et al. [19], where $k = \max_{i,j} r_{ij}$ and $\mathcal{H}(k) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}$. By setting $r_{ij} := \lceil F_G(i,j)/t \rceil$ for each pair of vertices i, j, our Light Edge-Connectivity–Spanner problem (with given flow–stretch factor t) can be reduced to SNDP.

Another related problem, which deals with the maximum flow, is investigated in [14, 25]. In that problem, called *MaxFlowFixedCost*, given a graph $G = (V, E)$ with nonnegative capacities $c(e)$ and nonnegative costs $p(e)$ for each edge $e \in E$, a source s and a sink t, and a positive number B, one must find an edge subset $E' \subseteq E$ of total cost $\sum_{e \in E'} p(e) \leq B$, such that in spanning graph $H = (V, E')$ of G the flow from s to t is maximized. Paper [14] shows that this problem, even with uniform edge-prices, does not admit a $2^{\log^{1-\epsilon} n}$-ratio approximation for any constant $\epsilon > 0$ unless $NP \subseteq DTIME(n^{polylog\, n})$. In [25], a polynomial time F^*-approximation algorithm for the problem is presented, where F^* denotes the maximum total flow. In our Sparse Flow–Spanner problem we require from spanning subgraph H to approximate maximum flows for all vertex pairs simultaneously.

To the best of our knowledge our spanner-like all-pairs problem formulations are new.

4. HARDNESS OF THE FLOW–SPANNER PROBLEMS

This section is devoted to the proof of the NP-completeness of the Sparse Edge-Connectivity–Spanner problem and other Flow–Spanner problems.

Theorem 1. *Sparse Edge-Connectivity–Spanner problem is NP-complete.*

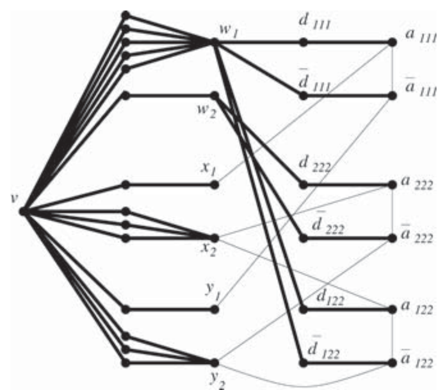

FIG. 1. Graph created according to 3DM instance: $M = \{(w_1, x_1, y_1), (w_2, x_2, y_2), (w_1, x_2, y_2)\}$, $W = (w_1, w_2), X = (x_1, x_2)$ and $Y = (y_1, y_2)$. The edges from E_d are shown in bold.

Proof. It is obvious that the problem is in NP. To prove its NP-hardness, we will reduce the 3-dimensional matching (3DM) problem to this one, by extending a reduction idea from [17].

Let $M \subseteq W \times X \times Y$ be an instance of 3DM, with $|M| = p$ and $W = \{w_i | i = 1, 2, \ldots, q\}, X = \{x_i | i = 1, \ldots, q\}$ and $Y = \{y_i | i = 1, \ldots, q\}$ (note that the sets W, X, Y are pairwise disjoint). One needs to check if M contains a matching, that is, a subset $M' \subseteq M$ such that $|M'| = q$ and no two triples of M' share a common element from $W \cup X \cup Y$.

Define $Deg(a)$ to be the number of triples in M that contain $a, a \in W \cup X \cup Y$. We construct a graph $G = (V, E)$ as follows (see Fig. 1). For each triple $(w_i, x_j, y_k) \in M$, there are four corresponding vertices $a_{ijk}, \overline{a}_{ijk}, d_{ijk}$ and $\overline{d}_{ijk}$ in V. d_{ijk} and $\overline{d}_{ijk}$ are called dummy vertices. Denote **F1**

$$D := \{d_{ijk} | (w_i, x_j, y_k) \in M\}, \quad \overline{D} := \{\overline{d}_{ijk} | (w_i, x_j, y_k) \in M\},$$

$$A := \{a_{ijk} | (w_i, x_j, y_k) \in M\}, \quad \overline{A} := \{\overline{a}_{ijk} | (w_i, x_j, y_k) \in M\}.$$

Additionally, for each $a \in X \cup Y$, we define a vertex a and $2Deg(a) - 1$ dummy vertices $d_1(a), \cdots, d_{2Deg(a)-1}(a)$ of a. For each $w_i \in W$, we define a vertex w_i and $4Deg(w_i) - 3$ dummy vertices $d_1(w_i), \cdots, d_{4Deg(w_i)-3}(w_i)$ of w_i. There is an extra vertex v in V. Let N_d be the dummy vertices (note that $D, \overline{D} \subset N_d$). The vertex set V of G is

$$V = \{v\} \cup W \cup X \cup Y \cup A \cup \overline{A} \cup N_d.$$

For each dummy vertex $d_i(a) \in N_d$ ($a \in W \cup X \cup Y$) put $(a, d_i(a)), (v, d_i(a))$ into E_d. Also put $(w_i, d_{ijk}), (d_{ijk}, a_{ijk}), (w_i, \overline{d}_{ijk}), (\overline{d}_{ijk}, \overline{a}_{ijk})$ into E_d. Now, the edge set E of G is

$$E = E_d \cup \{(a_{ijk}, \overline{a}_{ijk}), (a_{ijk}, x_j), (\overline{a}_{ijk}, y_k) | (w_i, x_j, y_k) \in M\}.$$

This completes the description of $G = (V, E)$. Clearly, each dummy vertex has exactly two neighbors in G, and each vertex of $A \cup \overline{A}$ has exactly 3 neighbors in G. Also, each vertex w_i has $4Deg(w_i) - 3 + 2Deg(w_i) = 6Deg(w_i) - 3$ neighbors in G, each vertex $a \in X \cup Y$ has $2Deg(a) - 1 + Deg(a) = 3Deg(a) - 1$ neighbors in G.

Set $t = 3/2$ and $B = |E_d| + p + q$. We claim that M contains a matching M' if and only if G has a flow–spanner $H = (V, E')$ with flow–stretch factor $\leq t$ and B edges.

Suppose M contains a matching M'. Add E_d to E'. For each triple $(w_i, x_j, y_k) \in M'$, put (a_{ijk}, x_j), $(\overline{a}_{ijk}, y_k)$ into E'. Because $|M'| = q$, the number of edges added to E' is $2q$. For each triple $(w_i, x_j, y_k) \in M \setminus M'$, put $(a_{ijk}, \overline{a}_{ijk})$ into E'. This will add $p - q$ edges into E'. Therefore E' contains $|E_d| + 2q + (p - q) = |E_d| + p + q$ edges. It is easy also to show that $fs_H \leq 3/2$, i.e., for any two vertices $s, t \in V(G)$, *flow−stretch*(s, t) is at most 3/2 (the technical details can be found in the extended version of this paper [11]).

Assume now that G has a flow–spanner $H = (V, E')$ with flow–stretch factor $\leq 3/2$ and with $B = |E_d| + p + q$ edges. First notice that E_d must be a subset of E' (otherwise, the flow–stretch factor of H is at least 2, contradicting our assumption). It is rather straightforward also to show that, for any vertex $u \in V' := X \cup Y \cup A \cup \overline{A}$, at least one edge $e \in E' \setminus E_d$ must be incident on u (the technical details can be found in the extended version of this paper [11]). Now, since $|V'| = 2p + 2q$ and there are $p + q$ edges in $E' \setminus E_d$, we conclude that, for each vertex $u \in V'$, exactly one edge from $E' \setminus E_d$ incident on it. Consequently, each x_j will have one edge (x_j, a_{ijk}) from $E' \setminus E_d$ incident on it. Hence, $(a_{ijk}, \overline{a}_{ijk})$ is not in E', and thus $(\overline{a}_{ijk}, y_k)$ must be in E'. No other edge in $E' \setminus E_d$ will be incident on y_k. Therefore, for each x_j, the corresponding triple (w_i, x_j, y_k) can be put in matching M'. The remaining $p - q$ edges in $E' \setminus E_d$ will be of the form $(a_{ijk}, \overline{a}_{ijk})$, and thus contribute nothing to the matching. This shows that M contains a matching M', completing the proof of the theorem. ∎

This theorem immediately implies the following corollary.

Corollary 1. *The Light Flow–Spanner, the Sparse Flow–Spanner and the Light Edge-Connectivity–Spanner problems are NP-complete.*

5. TREE FLOW–SPANNERS

In this section, we show that the Light Tree Flow–Spanner problem is NP-complete, whereas the Tree Flow–Spanner problem can be solved efficiently by any Maximum Spanning Tree algorithm.

Theorem 2. *The Light Tree Flow–Spanner problem is NP-complete.*

Proof. The problem is obviously in NP. One can nondeterministically choose a spanning tree and test in polynomial time whether it satisfies the cost and the flow–stretch bounds.

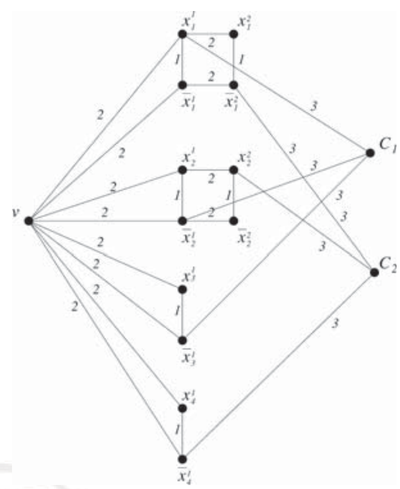

FIG. 2. Graph created from expression $(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_4)$.

To prove its NP-hardness, we will reduce the 3SAT problem to this one.

Let x_i be a variable in the 3SAT instance. Without loss of generality, assume that the 3SAT instance does not have clause of type $(x_i \vee \overline{x}_i \vee x_j)$ (note j may be equal to i). Since such a clause is always true, no matter what value x_i gets, it can be eliminated without affecting the satisfiability.

From a 3SAT instance one can construct a graph $G = (V, E)$ as follows. Let $x_1, x_2, \cdots, x_n$ be the variables and $C_1, \cdots, C_q$ be the clauses of 3SAT. Let k_i be the number of clauses containing either literal x_i or literal $\overline{x}_i$. Create a ladder in G on $2k_i$ vertices for each variable x_i in the following way. Create vertices $V(x_i) = \{x_i^1, x_i^2, \cdots, x_i^{k_i}\}$ and $\overline{V}(x_i) = \{\overline{x}_i^1, \cdots, \overline{x}_i^{k_i}\}$. All these vertices are called variable vertices. Put an edge $(x_i^l, \overline{x}_i^l)$ into $E(G)$, for $1 \leq l \leq k_i$. Set $p(x_i^l, \overline{x}_i^l) = c(x_i^l, \overline{x}_i^l) = 1$. For each integer l, where $1 \leq l < k_i$, put (x_i^l, x_i^{l+1}) and $(\overline{x}_i^l, \overline{x}_i^{l+1})$ into $E(G)$ and set their prices and capacities to 2.

For each clause C_j, create a clause vertex C_j in G. At the beginning, mark all variable vertices as "free." Do the following for $j = 1, 2, \ldots, q$ (in this order). If x_i (or $\overline{x}_i$) is in C_j, then find the smallest integer l such that x_i^l (or $\overline{x}_i^l$) is "free" and put (C_j, x_i^l) ($(C_j, \overline{x}_i^l)$, respectively) into $E(G)$. Mark x_i^l and $\overline{x}_i^l$ as "busy". Set $c(C_j, x_i^l) = p(C_j, x_i^l) = 3$ (respectively, $c(C_j, \overline{x}_i^l) = p(C_j, \overline{x}_i^l) = 3$).

Graph G has also one extra vertex v. For each variable x_i, put edges (v, x_i^1) and $(v, \overline{x}_i^1)$ into $E(G)$. Set their prices and capacities to 2. This completes the description of G. Obviously, the transformation can be done in polynomial time (Fig. 2). F2

It will be convenient to use the following notions. For each variable x_i, let H_i be the subgraph of G induced by vertices $\{v, x_i^1, \cdots, x_i^{k_i}, \overline{x}_i^1, \cdots, \overline{x}_i^{k_i}\}$. Name all the edges with capacity 2 assignment edges, the edges with capacity 1 connection edges and the edges with capacity 3 consistent edges. The path $(v, x_i^1, x_i^2, \cdots, x_i^{k_i})$ is called positive path of H_i and the path $(v, \overline{x}_i^1, \cdots, \overline{x}_i^{k_i})$ is called negative path of H_i.

Let $N = k_1 + k_2 + \cdots + k_n$. Set $B = 3N + 3q$ and $fs_T = 8$. We will show that the 3SAT is satisfiable if and only if the graph G has a tree flow–spanner with total cost less than or equal to B and flow–stretch factor at most 8.

Assume 3SAT is satisfiable. A tree flow–spanner T can be formed as follows. Put all the connection edges into T. For each variable x_i, if it is true, put all the edges in the positive path of H_i into $E(T)$, otherwise, put the edges in the negative path of H_i into $E(T)$. For each clause vertex C_j, identify one of its literals x_i ($\overline{x}_i$) which is true and put (C_j, x_i^l) ($(C_j, \overline{x}_i^l)$) into $E(T)$. Clearly, the number of connection edges put into $E(T)$ is $k_1 + k_2 + \cdots + k_n = N$. For each H_i, the number of assignment edges added into $E(T)$ is k_i. Hence, the total number of assignment edges added into $E(T)$ is $k_1 + \cdots + k_n = N$. The number of consistent edges added into $E(T)$ is q. From the above, one concludes that the total cost of T is $1 \times N + 2 \times N + 3 \times q = 3N + 3q = B$. Now, we need to show that for any two vertices $s, t \in V(G)$, *flow_ stretch*(s, t) is at most 8. We distinguish between 3 cases.

Case 1. *At least one of* $\{s, t\}$ *is a variable vertex.*

Assume, without loss of generality, that s is a variable vertex. Let $s = x_i^l$. By construction of G, x_i^l is incident to one connection edge, one or two assignment edges and at most one consistent edge. Hence, $F_G(s, t) \le 1 + 2 \times 2 + 3 = 8$ must hold. Since T is a spanning tree of G and every edge of G has capacity at least 1, $F_T(s, t)$ is at least 1. Therefore, *flow_ stretch*(s, t) is at most 8.

Case 2. *s is a clause vertex and t is v.*

Because s has exactly three consistent edges incident on it in G, $F_G(s, t) \le 9$. By construction of T, if a variable x_i is true (false), then the path between any vertex in $V(x_i)$ (in $\overline{V}(x_i)$, respectively) and v consists only of edges with capacity 2. Because s is attached to a vertex corresponding to a "true" literal by an edge with capacity 3, $F_T(s, t)$ is at least 2. This gives

$$flow{-}stretch(s, t) \le \frac{9}{2} = 4.5 < 8.$$

Case 3. *Both s and t are clause vertices.*

Let $s = C_i$ and $t = C_j$. We know that $F_G(s, t) \le 9$. Let $P_T(s, v), P_T(t, v)$ be the paths of T connecting v with s and t, respectively. Let $P_T(s, t)$ be the path between s and t in T. Clearly, $E(P_T(s, t)) \subseteq E(P_T(s, v)) \cup E(P_T(t, v))$. From the proof of Case 2, we have that any edge $e \in E(P_T(s, v)) \cup E(P_T(t, v))$ has capacity at least 2. Therefore, $F_T(s, t) \ge 2$ holds and *flow_ stretch*$(s, t) \le 9/2 = 4.5 < 8$ follows.

Thus, if 3SAT is satisfiable, then G has a tree flow–spanner with total cost B and flow–stretch factor 8. In what follows, we prove the "only if" direction.

Let T be a tree flow–spanner of G such that $fs_T \le 8$ and $\sum_{e \in E(T)} p(e) \le B$. Obviously, T must have at least q consistent edges. Assume T has r assignment edges, s connection edges and $t + q$ consistent edges. Clearly, $r, s, t \ge 0$ and, because T has $2N + q$ edges (because G has $2N + q + 1$ vertices), $r + s + t = 2N$. From $\sum_{e \in E(T)} p(e) \le B = 3N + 3q$ we conclude also that $2r + s + 3t \le 3N$. Hence, $2r + s + 3t - 2(r + s + t) \le -N$, i.e., $t \le s - N$. If $s < N$, then $t < 0$, which is impossible. Therefore, T must include all N connection edges of G, implying $s = N$ and $r + t = N$, $2r + 3t \le 2N$. From $2r + 3t - 2(r + t) \le 0$ we conclude that $t \le 0$. So, t must be 0, and therefore, T contains exactly q consistent edges, exactly N assignment edges and all N connection edges. This implies that, for every variable x_i, exactly one edge from $\{(x_i^1, v), (\overline{x}_i^1, v)\}$ is in $E(T)$. Because in T each clause vertex must be adjacent to at least one variable vertex and there are q consistent edges in T, each clause vertex is a pendant vertex of T (is adjacent in T to exactly one variable vertex). By construction of G, for each variable vertex x_i^l, any path between x_i^l and v in G either totally lies in H_i or has to use at least one clause vertex. Because all clause vertices are pendant in T, the path between x_i^l and v in T must totally lie in H_i. Similarly, the path between $\overline{x}_i^l$ and v in T must totally lie in H_i.

We show now how to assign true/false to the variables of the 3SAT instance to satisfy all its clauses. For each variable x_i, if $(x_i^1, v) \in E(T)$ then assign true to x_i, otherwise assign false to x_i. We claim that, if a clause vertex C_j is adjacent to a variable vertex x_i^l (or to a variable vertex $\overline{x}_i^l$) in T, then x_i is assigned true (false, respectively). The claim can be proved by contradiction. Assume x_i is assigned false, i.e., $(\overline{x}_i^1, v) \in E(T)$ and $(x_i^1, v) \notin E(T)$, but C_j is adjacent to a variable vertex x_i^l in T. As it was mentioned in the previous paragraph, the path $P_T(x_i^l, v)$ between x_i^l and v in T must totally lie in H_i. Because $(x_i^1, v) \notin E(T)$, edge (x_i^1, v) cannot be in $P_T(x_i^l, v)$. By construction of H_i, any path in H_i from x_i^l to v not using edge (x_i^1, v) must contain at least one connection edge. This means that the path $P_T(C_j, v)$ contains at least one connection edge, too. Because all connection edges have capacity 1, $F_T(C_j, v) = 1$. On the other hand, $F_G(C_j, v) = 9$. Hence, flow-stretch$(C_j, v) = 9 > 8$, contradicting with $fs_T \le 8$. This contradiction proofs the claim. Now, because every clause contains at least one true literal (note $(x_i^l, C_j) \in E(G)$ implies clause C_j contains x_i), the 3SAT instance is satisfiable.

This completes the proof of the theorem. ∎

Let $G = (V, E)$ be graph of an instance of the Light Tree Flow–Spanner problem. Let c^* be the maximum edge capacity of G and c_* be the minimum edge capacity of G. Note that, if $\frac{c^*}{c_*} = 1$, then the Light Tree Flow–Spanner problem can be solved in polynomial time by simply finding a minimum spanning tree T_p of G, where the weight of an edge $e \in E(G)$ is $p(e)$. From the proof of Theorem 2, one concludes that when $\frac{c^*}{c_*} \geq 3$, the Light Tree Flow–Spanner problem is NP-complete.

We turn now to the Tree Flow-Spanner problem on a graph $G = (V, E)$ (recall that in this problem $p(e) = 1$ for any $e \in E$). Let T_c be a maximum spanning tree of G, where the weight of an edge $e \in E(G)$ is $c(e)$. In what follows we show that the tree T_c is an optimal tree flow–spanner of G.

Lemma 1. *Let T_c be a maximum spanning tree of a graph G (with edge weights $c(\cdot)$) and T be an arbitrary spanning tree of G. Then, for any two vertices $u, v \in V(G)$, the following inequality holds,*

$$F_{T_c}(u, v) \geq F_T(u, v).$$

Proof. Let $u, v \in V(G)$ be two arbitrary vertices of G. Let $P_{T_c}(u, v)$ be the path connecting u and v in T_c and $P_T(u, v)$ be the path connecting u and v in tree T. Let $e_{u,v} \in P_{T_c}(u, v)$ and $e'_{u,v} \in P_T(u, v)$ be edges with minimum capacities in corresponding paths. To prove the lemma, one needs to show that $c(e_{u,v}) \geq c(e'_{u,v})$. If $P_{T_c}(u, v) = P_T(u, v)$, then the lemma trivially holds. Hence, we may assume that those paths do not coincide. We distinguish between two cases.

Case 1. *$P_{T_c}(u, v)$ and $P_T(u, v)$ are vertex-disjoint paths of G, i.e., they share only vertices u and v.*

Assume $c(e_{u,v}) < c(e'_{u,v})$. Let T_1, T_2 be two subtrees of T_c obtained from T_c by removing the edge $e_{u,v}$. Because $u \in T_1$ and $v \in T_2$, there must exist an edge $e' = (a, b) \in P_T(u, v)$ such that $a \in V(T_1)$ and $b \in V(T_2)$. By the choice of $e'_{u,v}$, the inequality $c(e') \geq c(e'_{u,v}) > c(e_{u,v})$ holds. Let T' be a spanning tree of G obtained from T_c by replacing the edge $e_{u,v}$ with edge e'. We get

$$\sum_{e \in E(T')} c(e) - \sum_{e \in E(T_c)} c(e) = c(e') - c(e_{u,v}) > 0,$$

and therefore the total weight of T' is greater than the total weight of T_c, contradicting with T_c being a maximum spanning tree of G. Thus, $c(e_{u,v}) \geq c(e'_{u,v})$ must hold.

Case 2. *$P_{T_c}(u, v)$ and $P_T(u, v)$ have some vertices in common different from u and v.*

We can decompose paths $P_{T_c}(u, v)$ and $P_T(u, v)$ into subpaths $P_1, P_2, \cdots, P_{2k+1}$ and $P'_1, P'_2, \cdots, P'_{2k+1}$ such that $\{P_i : i = 1, \ldots, 2k+1\}$ are subpaths of $P_{T_c}(u, v)$, $\{P'_i : i = 1, \ldots, 2k+1\}$ are subpaths of $P_T(u, v)$, P_i coincides with P'_i for all odd is, and subpaths P_i and P'_i are vertex-disjoint for all even is. Notice that some P_is (P'_is) can consist only of one vertex. Let $e_i \in P_i$ and $e'_i \in P'_i$ be edges such that $c(e_i), c(e'_i)$ are minimum among all the edges on P_i and P'_i, respectively. By the definition of $e_{u,v}$ and $e'_{u,v}$, we know $e_{u,v} \in \{e_1, \cdots, e_{2k+1}\}$ and $e'_{u,v} \in \{e'_1, \cdots, e'_{2k+1}\}$. Assume $e_{u,v} \in P_i$ and $e'_{u,v} \in P'_j$. From the discussion above we conclude that $c(e_{u,v}) = c(e_i) \geq c(e'_i)$. Since $c(e'_{u,v})$ is the minimum capacity of edges on $P_T(u, v)$, we deduce $c(e'_{u,v}) \leq c(e'_i)$. Combining the two above inequalities, we obtain $c(e'_{u,v}) \leq c(e_{u,v})$.

This concludes our proof. ∎

Lemma 1 implies that a maximum spanning tree T_c of a graph G, where the edge capacities are interpreted as edge weights, is an optimal tree flow–spanner of G. Hence, the following theorem holds.

Theorem 3. *Given an undirected graph $G = (V, E)$, with nonnegative capacities on edges, and a number $t > 0$, whether G admits a tree flow–spanner with flow–stretch factor at most t can be determined in polynomial time (by any maximum spanning tree algorithm).*

6. APPROXIMATION ALGORITHMS FOR THE LIGHT TREE FLOW–SPANNER PROBLEM

In this section, we present some approximation algorithms for the Light Tree Flow–Spanner problem. Let $G = (V, E)$ be an undirected graph with nonnegative edge capacities $c(e)$ and nonnegative edge costs $p(e)$, $e \in E(G)$. For given two positive numbers t and B we want to check if a spanning tree T^* of G with flow–stretch factor $fs_{T^*} \leq t$ and total cost $\mathcal{P}(T^*) \leq B$ exists or not. If such a tree exists then we say that the Light Tree Flow–Spanner problem on G has a solution. We will say that a spanning tree T of a graph G gives an (α, β)-approximate solution to the Light Tree Flow–Spanner problem on G if the inequalities $fs_T \leq \alpha t$ and $\mathcal{P}(T) \leq \beta B$ hold for T. A polynomial time algorithm producing an (α, β)-approximate solution to any instance of the Light Tree Flow–Spanner problem admitting a solution is called an (α, β)-approximation algorithm for the Light Tree Flow–Spanner problem.

One can easily see that the following lemma holds.

Lemma 2. *If $\frac{c^*}{c_*} \leq k$, where $c^* := max\{c(e) : e \in E\}$ and $c_* := min\{c(e) : e \in E\}$, then there is a $(k, 1)$-approximation algorithm for the Light Tree Flow–Spanner problem.*

Proof. Let $G = (V, E)$ be graph of an instance of the Light Tree Flow–Spanner problem. Interpret costs $p(e)$ as edge weights and construct a minimum weight spanning tree T_p of G. We claim that if the Light Tree Flow–Spanner problem on G has a solution, then T_p gives a $(k, 1)$-approximate solution to the problem. Indeed, let T^* be a solution to the Light Tree Flow–Spanner problem. Clearly, $\mathcal{P}(T_p) \leq \mathcal{P}(T^*)$. Consider two arbitrary vertices $u, v \in V(G)$. Because $F_{T^*}(u, v) \leq c^*$ and $F_{T_p}(u, v) \geq c_*$, from $F_G(u, v)/F_{T^*}(u, v) \leq t$ we have $F_G(u, v) \leq tF_{T^*}(u, v) \leq tc_*c^*/c_* \leq ktc_* \leq ktF_{T_p}(u, v)$. ∎

This result will be used in our main approximation algorithm. Let $G = (V, E)$ be an undirected graph with non-negative edge capacities $c(e)$ and non-negative edge costs

$p(e)$, $e \in E(G)$. Assume that G has a spanning tree T^* with $fs_{T^*} \leq t$ and $\mathcal{P}(T^*) \leq B$. In what follows, we describe a polynomial time algorithm which, given a parameter (any real number) r larger than 1 and smaller than t, produces a spanning tree T of G such that $fs_T \leq r(t-1)t$ and $\mathcal{P}(T) \leq 1.55 \log_r(r(t-1))B$ (note that the constant 1.55 comes from the approximation ratio for the Steiner Tree problem [32]). Thus, it is an $(r(t-1), 1.55\log_r(r(t-1)))$-approximation algorithm for the Light Tree Flow–Spanner problem. The parameter r of the algorithm can be chosen from the real interval $(1, t)$ by the user. If r is chosen to be equal to 2 then we have an $(2(t-1), 1.55\log_2(2(t-1)))$-approximation algorithm for the Light Tree Flow–Spanner problem. If $r = t-1$, then we get $((t-1)^2, 3.1)$-approximation algorithm.

Assume that the edges of G are ordered in a nondecreasing order of their capacities, i.e., we have an ordering $e_1, e_2, \cdots, e_m$ of the edges of G such that $c(e_1) \leq c(e_2) \cdots \leq c(e_m)$. Let $1 < r \leq t-1$. If $c(e_m)/c(e_1) \leq r(t-1)$, then Lemma 2 suggests to construct a minimum spanning tree of G using $p(e)$s as the edge weights. This tree is an $(r(t-1), 1)$-approximate solution, and hence we are done. Assume now that $c(e_m)/c(e_1) > r(t-1)$. We cluster all the edges of G into groups as follows. First group consists of all the edges whose capacities are in the range $[l_1 = c(e_m)/r, h_1 = c(e_m)]$. Then, we find the largest capacity $c(e_i)$ such that $c(e_i) < c(e_m)/r$ and form the second group of edges. It consists of all edges whose capacities are in the range $[l_2 = c(e_i)/r, h_2 = c(e_i)]$. We continue this process until a group of edges whose capacities are in the range $[l_k, h_k]$ with $c(e_1) \geq l_k$ is formed.

Let $G_i = (V, E_i)$ be a subgraph of G formed by $E_i = \{e \in E(G) : l_i \leq c(e) \leq h_1\}$. Let $G_1^i, G_2^i, \cdots, G_{p_i}^i$ be those connected components of G_i which contain at least two vertices. Consider another subgraph $G_i' = (V, E_i')$ of G formed by $E_i' = \{e \in E(G) : h_i/(r(t-1)) \leq c(e) \leq h_1\}$. $G_1'^i, G_2'^i, \cdots, G_{q_i}'^i$ are used to denote those connected components of G_i' which contain at least two vertices. Clearly, $E_i \subseteq E_{i+1}$ for any i.

Let $u, v \in V(G)$ be two arbitrary vertices. Choose the minimum i such that u and v are connected in G_i and let G_j^i be the connected component of G_i which contains u and v. Let $G_{j'}'^i$ be the connected component of G_i' such that $G_j^i \subseteq G_{j'}'^i$ (clearly, such a connected component exists). The following lemma holds.

Lemma 3. *If G has a tree flow–spanner T^* with flow–stretch factor $\leq t$, then the path $P_{T^*}(u,v)$ connecting u and v in T^* must totally lie in $G_{j'}'^i$.*

Proof. Proof is by contradiction. Assume the lemma is not true. Then we can find an edge e in $P_{T^*}(u,v)$ such that $c(e) < h_i/(r(t-1)) = l_i/(t-1)$. Because u and v are from G_j^i, there must exist two vertices $u', v' \in V(G_j^i) \cap V(P_{T^*}(u,v))$ such that the subpath $P_{T^*}(u', v')$ of $P_{T^*}(u,v)$ between u' and v' shares with G_j^i only u' and v' and e is an edge of $P_{T^*}(u', v')$. Because $u', v' \in G_j^i$, we get $F_G(u', v') \geq l_i + c(e)$. But then

$$\frac{F_G(u',v')}{F_{T^*}(u',v')} \geq \frac{l_i + c(e)}{c(e)} = \frac{l_i}{c(e)} + 1 > \frac{l_i}{l_i/(t-1)} + 1 = t.$$

This is in a contradiction with T^* being a tree t-flow–spanner of G. ∎

From Lemma 3, our approximation algorithm for the Light Tree Flow–Spanner problem is obvious.

Procedure 1. Construct a light tree flow–spanner for a graph *G*.

Input An undirected graph G with non-negative edge capacities $c(e)$ and non-negative edge costs $p(e)$, $e \in E(G)$; positive real numbers t and $1 < r \leq t-1$.

Output A spanning tree T of G.

Method
set $G_f := (V, E_f)$, where $E_f = \{e \in E(G) : p(e) = 0\}$;
for $i = 1$ **to** k **do**
let $G_i := (V, E_i)$ be a subgraph of G formed by $E_i := \{e \in E(G) : l_i \leq c(e) \leq h_1\}$;
let $G_1^i, G_2^i, \cdots, G_{p_i}^i$ be those connected components of G_i which contain at least two vertices;
let $G_i' := (V, E_i')$ be a subgraph of G formed by $E_i' := \{e \in E(G) : h_i/(r(t-1)) \leq c(e) \leq h_1\}$;
let $G_1'^i, G_2'^i, \cdots, G_{q_i}'^i$ be those connected components of G_i' which contain at least two vertices;
set $V_t := \bigcup_{1 \leq j \leq p_i} V(G_j^i)$;
in each connected component $G_j'^i$ $(1 \leq j \leq q_i)$, construct an approximate minimum weight Steiner tree $T_j'^i$ where terminals are $V(G_j'^i) \cap V_t$ and $p(e)$s are the edge weights;
set $E_f := E_f \bigcup \{\bigcup_{1 \leq j \leq q_i} \{e \in E(T_j'^i) : p(e) > 0\}\}$;
for each edge $e \in \bigcup_{1 \leq j \leq p_i} E(G_j^i)$, set $p(e) := 0$;
construct a maximum spanning tree T of G_f using the capacities as the edge weights;
return T.

Below, the quality of the tree flow–spanner T constructed by above procedure is analyzed.

Lemma 4. *If G admits a tree t-flow–spanner, then $fs_T \leq r(t-1)t$.*

Proof. Let $u, v \in V(G)$ be two arbitrary vertices and T^* be a tree t-flow–spanner of G. Choose the smallest integer i such that u and v are connected in G_i. Let $P_G(u,v)$ be an arbitrary path between u and v in G and $e \in P_G(u,v)$ be an edge on the path with smallest capacity. By the choice of i, we have $c(e) \leq h_i$.

Without loss of generality, assume $u, v \in G_j^i$. According to Procedure 1, u and v will be connected by a path $P_{T_j'^i}(u,v)$ in $T_j'^i$. Let $e' \in P_{T_j'^i}(u,v)$ be an edge with minimum capacity in $P_{T_j'^i}(u,v)$. It is easy to see that $c(e') \geq h_i/(r(t-1))$.

We claim that after iteration i, there is a path $P_{G_f}(u,v)$ between u and v in G_f such that for any edge $e \in P_{G_f}(u,v)$, the inequality $c(e) \geq h_i/(r(t-1))$ holds. We prove this claim by induction on i. All edges of $P_{T_j'^i}(u,v)$ with current $p(e)$ greater than 0 are added to E_f. E_f contains also each edge for which original $p(e)$ was 0. Therefore, if G_f does not contain an edge $e=(a,b) \in E(P_{T_j'^i}(u,v))$, then current $p(e)$ of e was 0, and this implies $c(e) > h_i$. According to Procedure 1, a and b must be in a connected component of G_l where $1 \leq l < i$. Hence, by induction, at lth iteration, a and b must be connected by a path $P_{G_f}(a,b)$ such that, for each edge $e \in P_{G_f}(a,b)$, the inequality $c(e) \geq h_l/(r(t-1)) > h_i/(r(t-1))$ holds. By concatenating such paths and the edges put into G_f during ith iteration, one can find a path between u and v which satisfies the claim.

Because T is a maximum spanning tree of G_f (where the edge weights are their capacities), similarly to the proof of Lemma 1, one can show that for any edge $e \in P_T(u,v)$, $c(e) \geq h_i/(r(t-1))$ holds. This implies $F_{T^*}(u,v) \leq h_i \leq r(t-1)F_T(u,v)$. Because T^* has flow–stretch factor $\leq t$, we have $F_G(u,v) \leq tF_{T^*}(u,v)$, and therefore

$$\frac{F_G(u,v)}{F_T(u,v)} \leq r(t-1)t.$$

This concludes our proof. ∎

The following lemma bounds the total cost of the tree flow–spanner T.

Lemma 5. *If G has a tree t-flow–spanner T^* with cost $\mathcal{P}(T^*)$, then $\mathcal{P}(T) \leq 1.55\log_r(r(t-1))\mathcal{P}(T^*)$.*

Proof. By Lemma 3, one knows that for any two vertices u, v of G_j^i, $P_{T^*}(u,v)$ totally lies in $G_{j'}'^i$ where $G_j^i \subseteq G_{j'}'^i$. Hence, the smallest subtree of T^* spanning all vertices of $V_t \cap G_{j'}'^i$ is totally contained in $G_{j'}'^i$. We can use in Procedure 1 an 1.55-approximation algorithm of Robins and Zelikovsky [32] to construct an approximation to a minimum weight Steiner tree in $G_{j'}'^i$ spanning terminals $V_t \cap V(G_{j'}'^i)$. It is easy to see that $\mathcal{P}_i(G_f) \leq 1.55\mathcal{P}_i(T^*)$, where $\mathcal{P}_i(G_f)$ is the total cost of the Steiner trees constructed by Procedure 1 on ith iteration and $\mathcal{P}_i(T^*)$ is the total cost of the edges from T^* which have capacities in the range $[h_i/(r(t-1)), h_i]$ and are used to connect vertices in V_t. Therefore, the following inequality holds:

$$\mathcal{P}(G_f) \leq \sum_{1\leq i\leq k} \mathcal{P}_i(G_f) \leq 1.55 \sum_{1\leq i\leq k} \mathcal{P}_i(T^*).$$

We will prove that

$$\sum_{1\leq i\leq k} \mathcal{P}_i(T^*) \leq \log_r(r(t-1))\mathcal{P}(T^*).$$

To see this, we show that each edge of T^* appears at most l times in $\sum_{1\leq i\leq k} \mathcal{P}_i(T^*)$, where

$$\frac{1}{r^l} \geq \frac{1}{r(t-1)}.$$

Then $l \leq \log_r(r(t-1))$ will follow.

Consider an edge $e \in G_i'$ with $p(e) \neq 0$. We have $h_i/(r(t-1)) \leq c(e) \leq h_i$. According to Procedure 1, after ith iteration, all the edges with capacity in $[h_i/r, h_i]$ have 0 cost. After $(i+1)$th iteration, all the edges with capacity in $[h_i/r^2, h_i]$ have 0 cost. After $(i+l-1)$th iteration, all the edges with capacity in $[h_i/r^l, h_i]$ have 0 cost. To have $p(e) > 0$, the inequality $h_i/r^l \geq h_i/(r(t-1))$ must hold. So, $l \leq \log_r(r(t-1))$ and therefore

$$\mathcal{P}(G_f) \leq 1.55\log_r(r(t-1))\mathcal{P}(T^*).$$

Because T is a spanning tree of G_f, the lemma clearly follows. ∎

Theorem 4. *There exists an $(r(t-1), 1.55\log_r(r(t-1)))$-approximation algorithm for the Light Tree Flow–Spanner problem, where r $(1 < r < t)$ is a parameter of the algorithm that can be chosen between 1 and t. If r is chosen to be equal to 2 then we have an $(2(t-1), 1.55\log_2(2(t-1)))$-approximation algorithm. If $r = t-1$, then we get $((t-1)^2, 3.1)$-approximation algorithm.*

In the remaining part, we describe how to get a tree flow–spanner T of G with flow–stretch factor $\leq t$ and total cost at most $(n-1)\mathcal{P}(T^*)$, provided G has a tree t-flow–spanner T^*. The algorithm is as follows.

Procedure 2. Construct a light tree t-flow–spanner for a graph G.

Input: An undirected graph G with non-negative edge capacities $c(e)$ and non-negative edge costs $p(e)$, $e \in E(G)$; a positive real number t.

Output: A tree t-flow–spanner T of G.

Method:

set $G_f := (V_f, E_f)$, where $V_f = V, E_f = \emptyset$;
construct a complete graph $G' = (V, E')$, where $E' = \{(u,v) : u, v \in V(G)$ and $u \neq v\}$;
for each $(u,v) \in E'$, let $w(u,v) := F_G(u,v)$ be the weight of the edge;
construct a maximum spanning tree T' of the weighted graph G';
for each edge $(u,v) \in E(T')$ **do**
  let $G_{w(u,v)}$ be a subgraph of G obtained from G by eliminating all the edges e such that $c(e) < w(u,v)/t$;
  find a connected component $G_{u,v}$ of $G_{w(u,v)}$ such that $u, v \in V(G_{u,v})$;
  if we cannot find such a connected component, then **return** "G does not have any flow tree t-spanner;"
  find a shortest (with respect to the costs of the edges) path $P_{G_{u,v}}(u,v)$ between u and v;
  set $E_f := E_f \cup E(P_{G_{u,v}}(u,v))$;
construct a maximum spanning tree T of G_f using the edge capacities as their weights;
return T.

The following lemmata are true.

Lemma 6. *The inequality* $\mathcal{P}(T) \leq (n-1)\mathcal{P}(T^*)$ *holds.*

Proof. If T^* is a tree t-flow–spanner of G, then for any two vertices u, v of G, the path $P_{T^*}(u, v)$ which connects u and v in T^* must use only edges of G with $c(e) \geq w(u, v)/t$.

Since for each edge $(u, v) \in E(T')$, Procedure 2 finds a shortest (with respect to the costs of the edges) path between u and v in $G_{u,v}$, the cost of this path is no more than $\mathcal{P}(T^*)$. T' has $n-1$ edges, so $\mathcal{P}(G_f) \leq (n-1)\mathcal{P}(T^*)$. Since T is a spanning tree of G_f, its cost is at most $\mathcal{P}(G_f)$. This gives $\mathcal{P}(T) \leq (n-1)\mathcal{P}(T^*)$. ∎

Lemma 7. *T has flow–stretch factor $\leq t$.*

Proof. To prove the lemma, one needs to show that for every edge $(u, v) \in E(G')$, the inequality $F_G(u, v) \leq tF_T(u, v)$ holds.

If $(u, v) \in E(T')$, then the inequality clearly holds. Assume $(u, v) \notin E(T')$. Let $P_{T'}(u, v)$ be the path between u and v in T'. Let (x, y) be an edge of $P_{T'}(u, v)$ such that $w(x, y)$ is minimum among all the edges on $P_{T'}(u, v)$. We claim $w(x, y) \geq w(u, v)$. Assume not. Then the tree $T'' = (T' \setminus \{(x, y)\}) \cup \{(u, v)\}$ will have larger weight than T', contradicting with T' being a maximum spanning tree of G'. Since for every edge $(u, v) \in E(G')$, $w(u, v) = F_G(u, v)$, we conclude $F_G(x, y) \geq F_G(u, v)$.

The above shows that for every edge $(x, y) \in E(P_{T'}(u, v))$, $F_G(x, y) \geq F_G(u, v)$ holds. Combining this with the fact that $F_G(x, y) \leq tF_T(x, y)$ for every edge $(x, y) \in E(T')$, we can easily show that for every edge $(u, v) \notin E(T')$, the inequality $F_G(u, v) \leq tF_T(u, v)$ still holds. Indeed, $F_G(u, v) \leq F_G(x, y) \leq tF_T(x, y)$ for every $(x, y) \in E(P_{T'}(u, v))$ and, therefore, $F_G(u, v) \leq t \min\{F_T(x, y) : (x, y) \in E(P_{T'}(u, v))\} = tF_T(u, v)$. ∎

Theorem 5. *There exists an $(1, n-1)$-approximation algorithm for the Light Tree Flow–Spanner problem.*

REFERENCES

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs, Discrete Comput Geom 9 (1993), 81–100.

[2] H.-J. Bandelt and A. Dress, Reconstructing the shape of a tree from observed dissimilarity data, Adv Appl Math 7 (1986), 309–343.

[3] S. Baswana and S. Sen, A simple linear time algorithm for computing a $(2k-1)$-spanner of $o(n^{1+1/k})$ size in weighted graphs, 30th International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science 2719, 2003, pp. 384–396.

[4] A. Brandstädt, F.F. Dragan, H.-O. Le, and V.B. Le, Tree spanners on chordal graphs: complexity and algorithms, Theoretical Comput Sci 310 (2004), 329–354.

[5] U. Brandes and D. Handke, *NP*–Completeness results for minimum planar spanners, Discrete Math Theoretical Comput Sci 3 (1998), 1–10.

[6] L. Cai and D.G. Corneil, Tree-spanners, SIAM J Discrete Math 8 (1995), 359–387.

[7] V.D. Chepoi, F.F. Dragan, and C. Yan, Additive spanners for k-chordal graphs, 5th Italian Conference on Algorithms and Complexity (CIAC), Vol. 2653, Lecture Notes in Computer Science, 2003, pp. 96–107.

[8] J. Cheriyan and R. Thurimella, Approximating minimum-size k-connected spanning subgraphs via matching, SIAM J Comput 30 (2000), 528–560.

[9] L.P. Chew, There are planar graphs almost as good as the complete graph, J Comput Sys Sci 39 (1989), 205–219.

[10] F.F. Dragan and C. Yan, Network flow spanners, Proc of the 7th Latin American Symposium LATIN 2006: Theoretical Informatics, Valdivia, Chile, March 20–24, Lecture Notes in Computer Science 3887, Springer, pp. 410–422.

[11] F.F. Dragan and C. Yan, Network flow spanners, Available at: http://www.cs.kent.edu/∼dragan/FlowSp-Full.pdf.

[12] M. Elkin and D. Peleg, $(1+\epsilon, \beta)$-spanner constructions for general graphs, 33rd Annual ACM Symposium on Theory of Computing (STOC) (2001), pp. 173–182.

[13] Y. Emek and D. Peleg, Approximating minimum max-stretch spanning trees on unweighted graphs, Proc of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004), New Orleans, Louisiana, January 11–14, 2004, pp. 261–270.

[14] G. Even, G. Kortsarz, and W. Slany, On network design problems: fixed cost flows and the Covering Steiner Problem, ACM Transactions on Algorithms 1 (2005), 74–101.

[15] S.P. Fekete and J. Kremer, Tree spanners in planar graphs, Discrete Appl Math 108 (2001), 85–103.

[16] C.G. Fernandes, A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem, J Algorithms 28 (1998), 105–124.

[17] G.N. Frederickson and J. JáJá, Approximation algorithms for several graph augmentation problems, SIAM J Comput 10 (1981), 270–283.

[18] H.N. Gabow, M.X. Goemans, E. Tardos, and D.P. Williamson, Approximating the smallest k-edge connected spanning subgraph by LP-rounding, Proc of the 16th Symposium on Discrete Algorithms (SODA 2005), 562–571.

[19] M.X. Goemans, A.V. Goldberg, S.A. Plotkin, D.B. Shmoys, É. Tardos, and D.P. Williamson, Improved approximation algorithms for network design problems, Proc of the 5th Symposium on Discrete Algorithms (SODA 1994), 223–232.

[20] H.N. Gabow, M.X. Goemans, and D.P. Williamson, An efficient approximation algorithm for the survivable network design problem, Math Program 82 (1998), 13–40.

[21] R.E. Gomory and T.C. Hu, Multi-terminal network flows, J SIAM 9 (1961), 551–570.

[22] R. Hassin and A. Levin, Minimum restricted diameter spanning trees, Proc 5th Int Workshop on Approximation Algorithms for Combinatorial Optimization, Lecture Notes in Computer Science 2462, Springer-Verlag, 2002, pp. 175–184. AQ1

[23] K. Jain, A factor 2-approximation algorithm for the generalized Steiner network problem, Combinatorica 21 (2001), 39–60.

[24] S. Khuller and U. Vishkin, Biconnectivity approximations and graph carvings, 24th Annual ACM Symposium on Theory of Computing (STOC) (1992), pp. 759–770.

[25] S.O. Krumke, H. Noltemeier, S. Schwarz, H.-C. Wirth, and R. Ravi, Flow improvement and network flows with fixed costs, Proc of the International Conference on Operations Research (OR'98), Springer, 1998 Available at: http://www.mathematik.uni-kl.de/pub/scripts/krumke/or98-flow.pdf.

[26] A.L. Liestman and T. Shermer, Additive graph spanners, Networks 23 (1993), 343–364.

[27] H. Nagamochi and T. Ibaraki, A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph, Algorithmica 7 (1992), 583–596.

[28] D. Peleg, Distributed computing: a locality-sensitive approach, SIAM Monographs Discrete Math Appl, SIAM, Philadelphia, 2000.

[29] D. Peleg and A.A. Schäffer, Graph spanners, J Graph Theory 13 (1989), 99–116.

[30] D. Peleg and J.D. Ullman, An optimal synchronizer for the hypercube, Proc 6th ACM Symposium on Principles of Distributed Computing, Vancouver, 1987, pp. 77–85.

[31] D. Peleg and E. Upfal, A tradeoff between space and efficiency for routing tables, 20th ACM Symposium on the Theory of Computing, Chicago, 1988, pp. 43–52.

[32] G. Robins and A. Zelikovsky, Improved steiner tree approximation in graphs, Proc 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), 770–779.

[33] J. Soares, Graph spanners: a survey, Congressus Numer 89 (1992), 225–238.

[34] G. Venkatesan, U. Rotics, M.S. Madanlal, J.A. Makowsky, and C. Pandu Ragan, Restrictions of minimum spanner problems, Informat Comput 136 (1997), 143–164.

[35] D.P. Williamson, M.X. Goemans, M. Mihail, and V.V. Vazirani, A primal-dual approximation algorithm for generalized Steiner network problems, Proc of the 25th Annual ACM Symposium on Theory of Computing (STOC 1993) (1993), pp. 708–717.