Network Flow Spanners

F. F. Dragan and Chenyu Yan

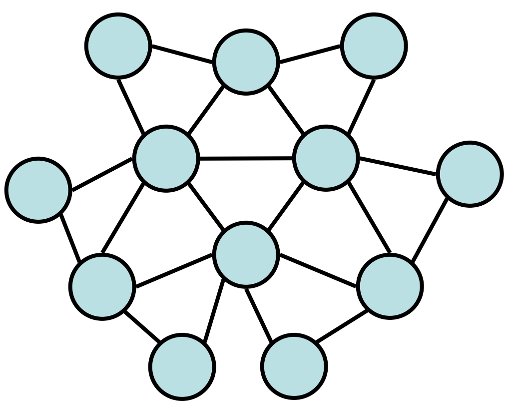
Kent State University, Kent, OH, USA

Well-known **Tree *t* -Spanner Problem**

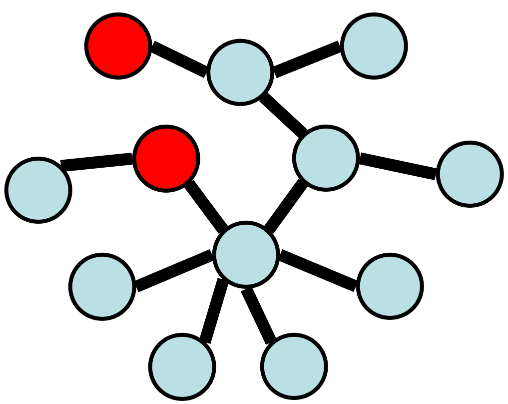
Multiplicative* Tree *t-Spanner:

- Given unweighted undirected graph $G=(V,E)$ and an integer t.
- Does G admit a spanning tree $T=(V,E')$ such that

$$\forall u, v \in V, \quad dist_T(v,u) \leq t \times dist_G(v,u) \ ?$$



G



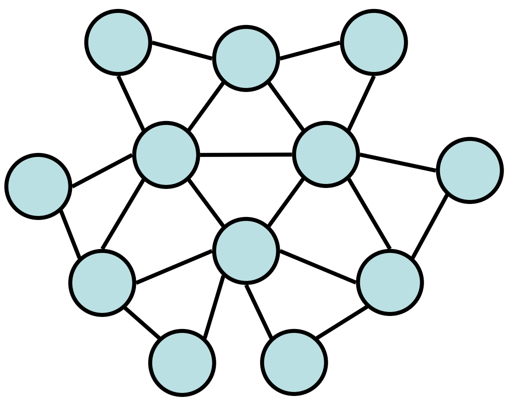
multiplicative tree 4-spanner of *G*

Well-known **Sparse *t*-Spanner Problem**

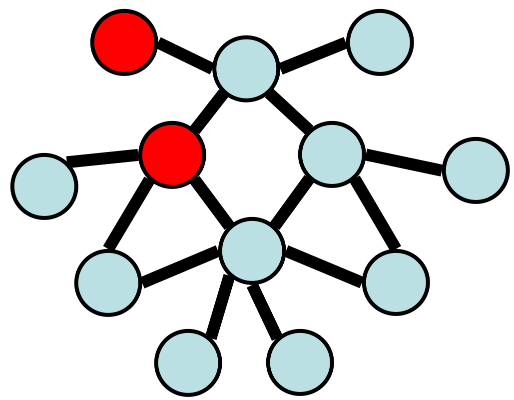
Multiplicative t-Spanner:

- Given unweighted undirected graph *G=(V,E)* and integers *t, m*.
- Does *G* admit a spanning graph *H =(V,E')* with |*E'*| $\leq m$ such that

$$\forall u,v \in V, \quad dist_H(v,u) \leq t \times dist_G(v,u) \ ?$$



G



multiplicative 2-spanner of *G*

New Light Flow-Spanner Problem

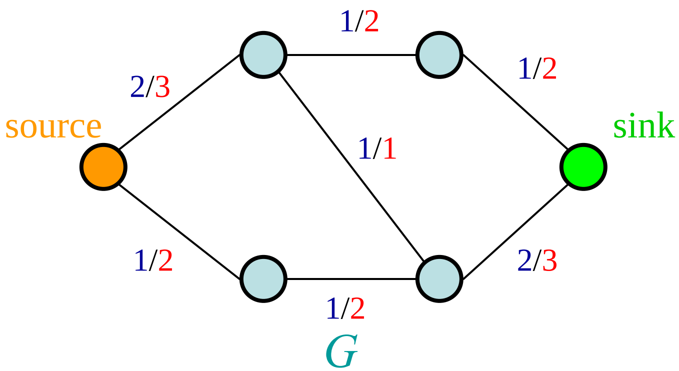
Light Flow-Spanner (LFS):

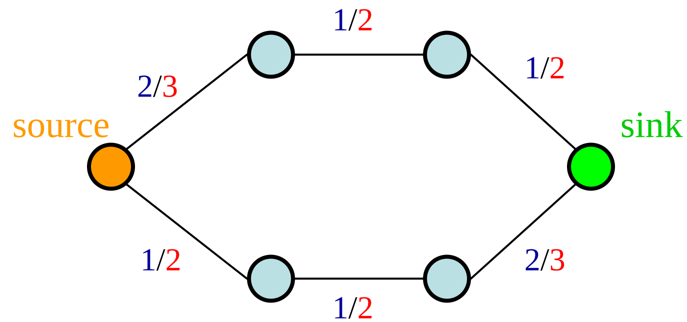
- Given undirected graph $G=(V,E)$, edge-costs $p(e)$ and edge-capacities $c(e)$, and integers B, t.

- Does G admit a spanning subgraph $H=(V,E')$ such that

$$\forall u,v \in V,\quad F_G(u,v) \le t \times F_H(u,v) \quad \text{and} \quad \sum_{e\in E'} p(e) \le B\ ?$$

($F_G(u, v)$ denotes the maximum flow between u and v in G.)





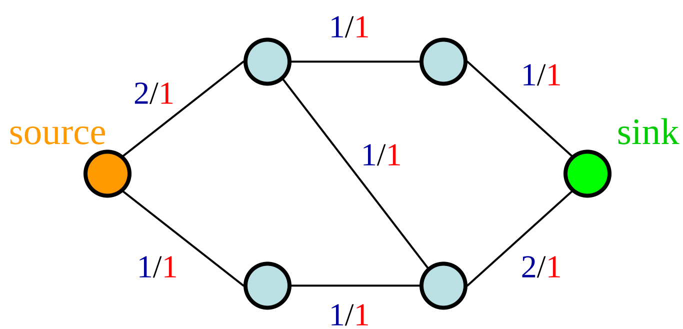
An LFS with flow stretch factor of 1.25 and budget 8

Variations of Light Flow-Spanner Problem

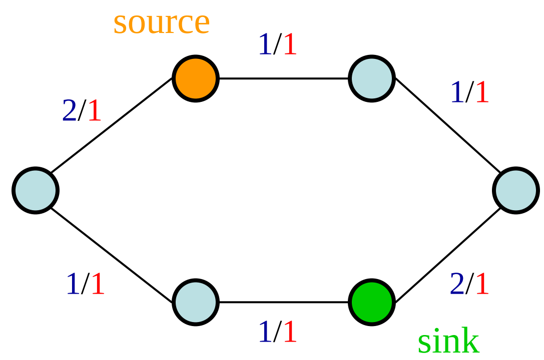
Sparse Flow-Spanner (SFS) : In the LFS problem, set $p(e)=1$, $e \in E$.

Sparse Edge-Connectivity-Spanner (SECS) : In the LFS problem, set $p(e)=1$, $c(e)=1$ for each $e \in E$.

Light Edge-Connectivity-Spanner (LECS) : In the LFS problem, for each $e \in E$ set $c(e)=1$.



G



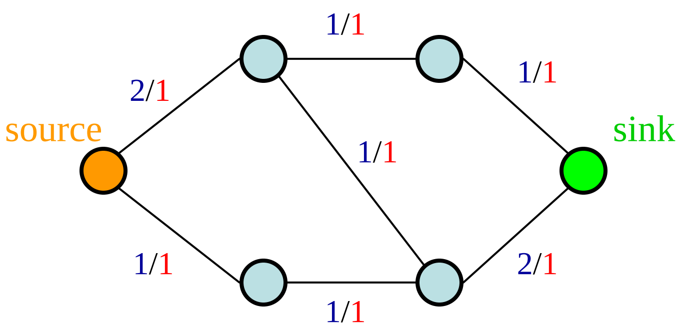
An LECS with flow stretch factor of 1.5 and budget 8

Variations of Tree Flow-Spanner Problem

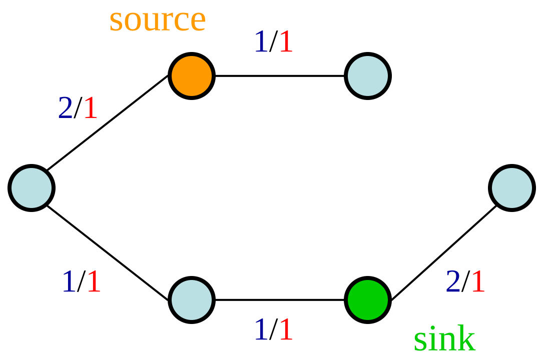
Tree Flow-Spanner (TFS) : In the LFS problem, we require the underlying spanning subgraph to be a tree and, for each $e \in E$, set $p(e)=1$.

➔ easy: max. spanning tree, capacities are the edge-weights

Light Tree Flow-Spanner (LTFS) : In the LFS problem, we require the underlying spanning subgraph to be a tree.



G



An LTFS with flow stretch factor of 3 and budget 7

Related Work

k-Edge-Connected-Spanning-Subgraph problem:

- Given a graph G along with an integer k, one seeks a spanning subgraph of G that is k-edge-connected
  - MAX SNP-hard [Fernandes'98]
  - (1+2/k)-approximation algorithm [Gabow et. al.'05]
  - Linear time with k|V| edges [Nagamochi&Ibaraki'92]

- Original edge-connectivities are not taking into account

Related Work

Survivable-network-design problem (SNDP):

- Given a 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.
  - NP-hard as a generalization of the Steiner tree problem
  - 2(1+1/2+1/3+…+1/k)-approximation algorithm [Gabow et. al.'98, Goemans et. al.'94]

- By setting $r_{ij}=\lceil F_G(i, j)/t \rceil$ for each pair of vertices i, j, our **Light Edge-Connectivity-Spanner** problem can be reduced to SNDP.

Related Work

MaxFlowFixedCost problem: [Krumke et. al.'98]

- Given a graph G, for every edge $e \in E$ a non-negative cost $p(e)$ and a non-negative capacity $c(e)$, a source s and a sink t, and a positive integer B. One needs to find a subgraph H of G of total cost at most B such that the maximum flow between s and t in H is maximized.
  - Hard to approximate
  - F^*-approximation algorithm (F^* is the maximum total flow)

- In our formulation, we approximate maximum flows for all vertex pairs simultaneously

Our results

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

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

- Two approximation algorithms for the ***Light Tree Flow-Spanner*** problem

SECS is NP-Complete

Sparse Edge-Connectivity-Spanner (SECS) is NP-hard

- Reduce 3-dimensional matching (3DM) to SECS.
- Let $M \subseteq W \times X \times Y$ be an instance of 3DM. For each element $a \in W \cup X \cup Y$, let *Deg(a)* be the number of triples in M that contains a.

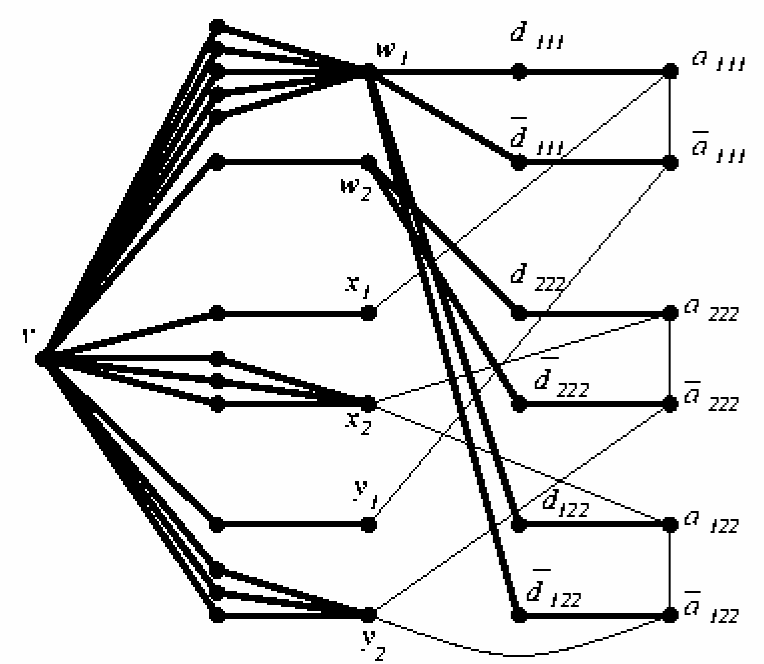
– For each triple $(w_i, x_j, y_k) \in M$, create four vertices a_{ijk}, $\bar{a}_{ijk}$, d_{ijk}, $\bar{d}_{ijk}$

– For each vertex $a \in X \cup Y$, create a vertex a and $2Deg(a)-1$ dummy vertices

– For each vertex $a \in W$, create a vertex a and $4Deg(a)-3$ dummy vertices

– Add one more vertex v and make connections ($E=E' \cup E''$)

– Set $\boldsymbol{t=3/2}$ and $\boldsymbol{B}=|\boldsymbol{M}|+|\boldsymbol{X}|+|\boldsymbol{E'}|$ (=3+2+…)



$M = \{(w_1, x_1, y_1), (w_2, x_2, y_2), (w_1, x_2, y_2)\}$

LTFS is NP-Complete

Light Tree Flow-Spanner (LTFS) ***is*** *NP-hard*

- Reduce 3SAT to LTFS. Let x_1, x_2, …, x_n be the variables and C_1, …, C_q the clauses of a 3SAT instance.

$$(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_4)$$

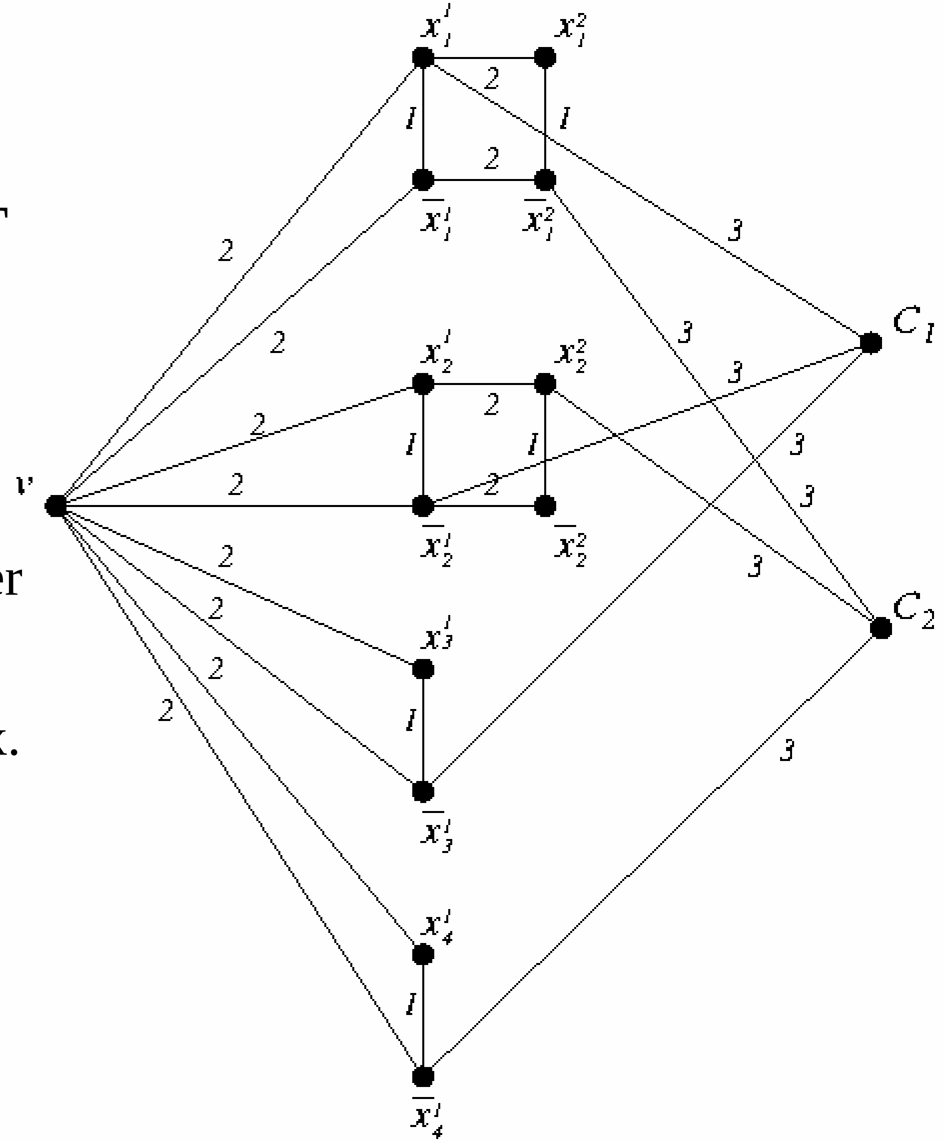
– For each variable x_i, create $2k_i$ vertices. k_i is the number of clauses containing either literal x_i or its negation.

– For each clause C_i create a clause vertex.

– Add one more vertex v.

– Add edges and set their capacities/costs

– Set $\boldsymbol{t=8}$ and $\boldsymbol{B=3(k_1+k_2+\ldots+k_n)+3q}$



NP-Completeness Results

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

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

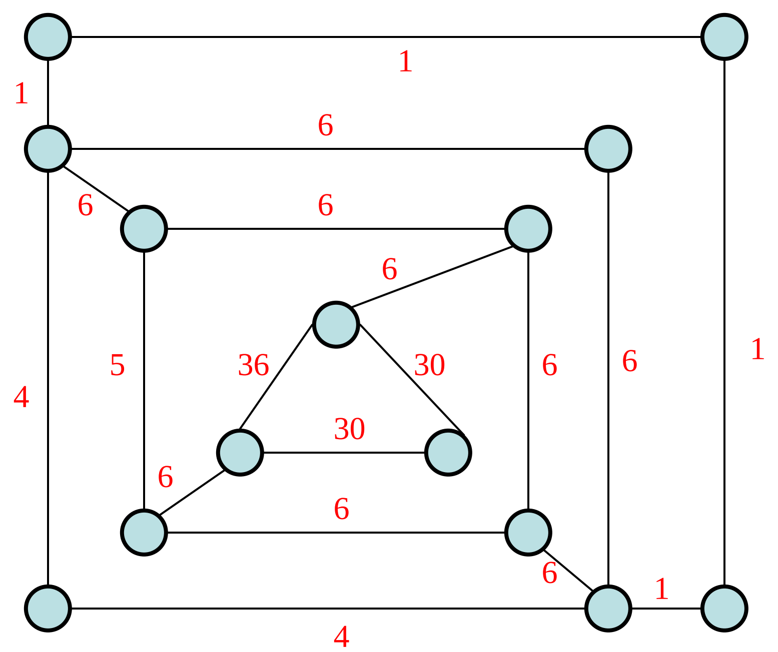
Theorem 1 immediately gives us the following corollary.

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

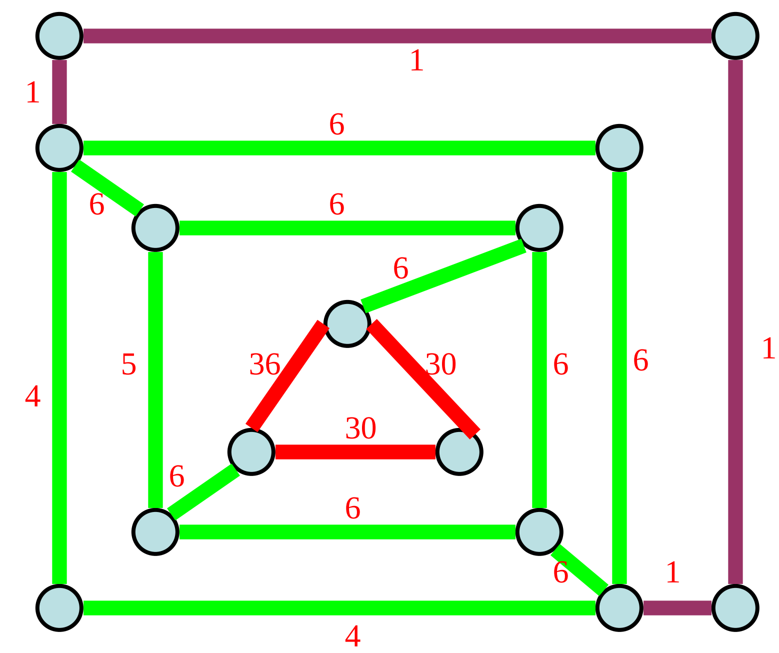
Approximation Algorithm for LTFS

Assume G has a ***Light Tree Flow-Spanner*** with flow-stretch factor t and budget B.

- Sort the edges of G such that $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$. Let $1 < r \leq t-1$.
- Cluster the edges according to the intervals $[l_k, h_k], \ldots, [l_1, h_1]$, where $h_1 = c(e_m)$ and $l_1 = h_1/r$ and, for $k \leq i < 1$, h_i is the largest capacity of the edge such that $h_i < l_{i-1}$, and $l_i = h_i/r$.



G

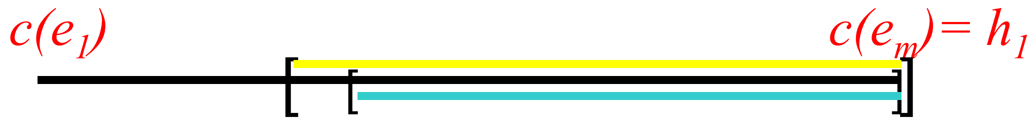


Clusters of edges of G $(r=2, t=3)$

[0.5,1] **[3, 6]** **[18, 36]**

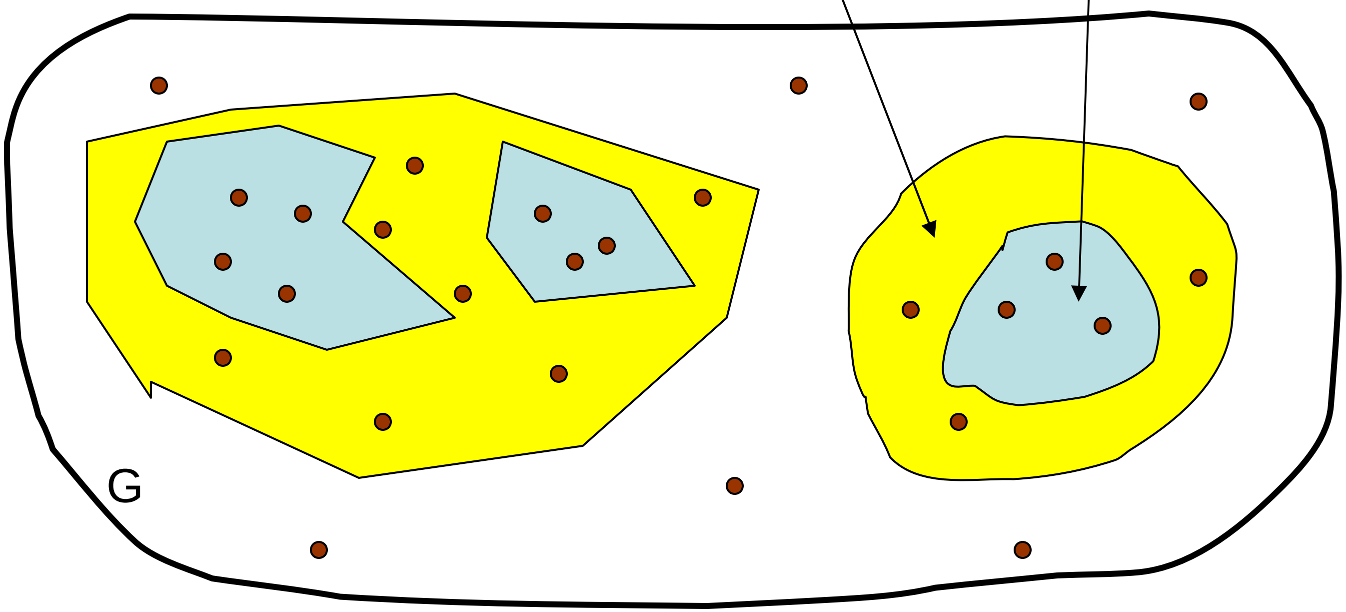
Approximation Algorithm for LTFS

General step:



Define $E_i^{'} = \{e \in E(G) : l_i/(t-1) \le c(e) \le h_1\}$ $E_i^{''} = \{e \in E(G) : l_i \le c(e) \le h_1\}$

- For each connected component of $G_i^{'} = (X_i, E_i^{'})$ construct a minimum weight Steiner-tree where the terminals are vertices from $G_i^{''} = (Y_i, E_i^{''})$ and the prices are the edge weights.
- Set the price of each edge in $E_i^{''}$ to 0. The Steiner-tree edges are stored in *F*.

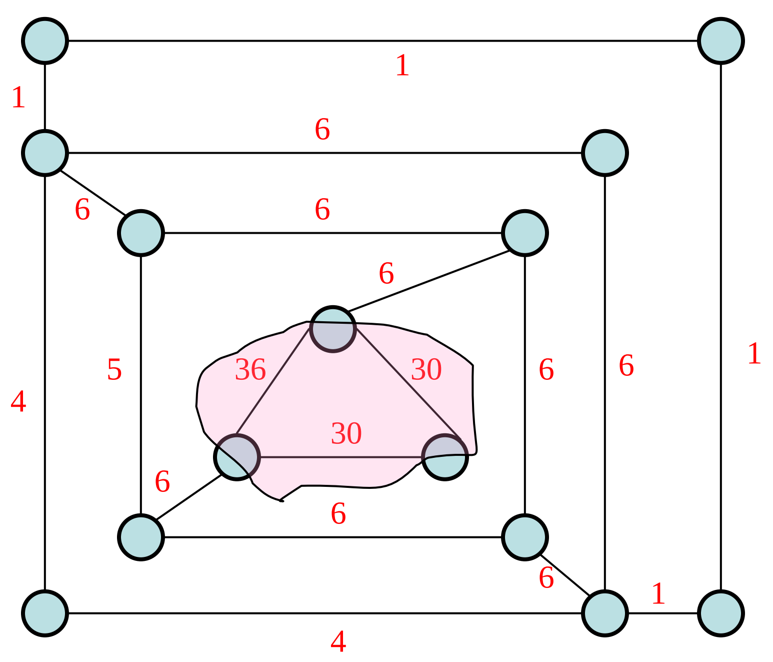


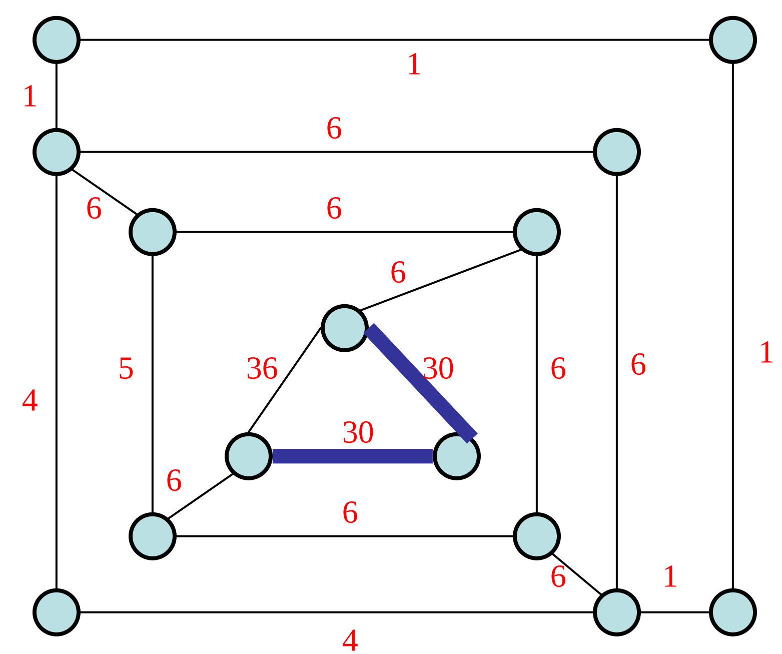
Approximation Algorithm for LTFS

[9, 36] [18, 36]

Define $E_i^{'} = \{e \in E(G) : l_i / (t-1) \leq c(e) \leq h_1\}$ $E_i^{''} = \{e \in E(G) : l_i \leq c(e) \leq h_1\}$

- For each connected component of $G_i^{'} = (X_i, E_i^{'})$ construct a minimum weight Steiner-tree where the terminals are vertices from $G_i^{''} = (Y_i, E_i^{''})$ and the prices are the edge weights.
- Set the price of each edge in $E_i^{''}$ to 0. The Steiner-tree edges are stored in F.



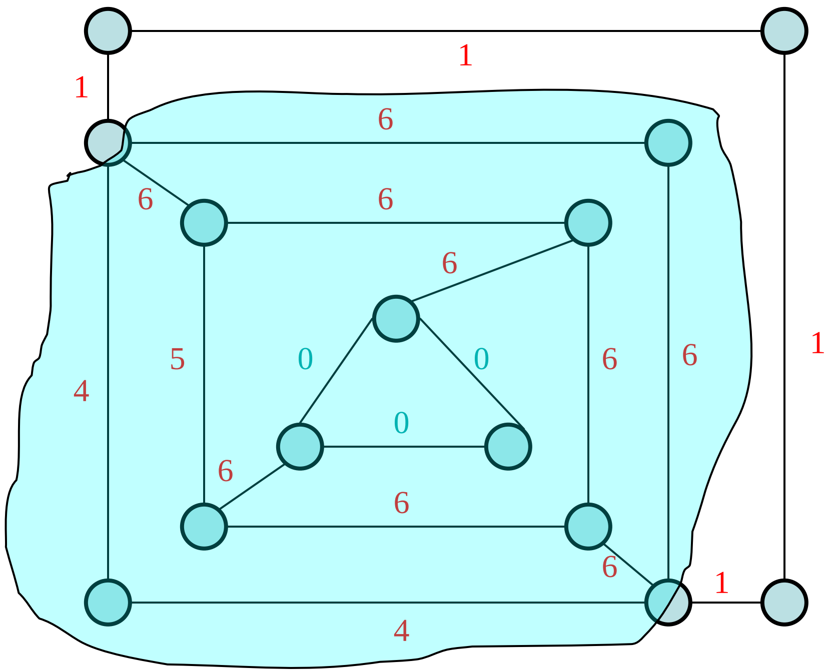


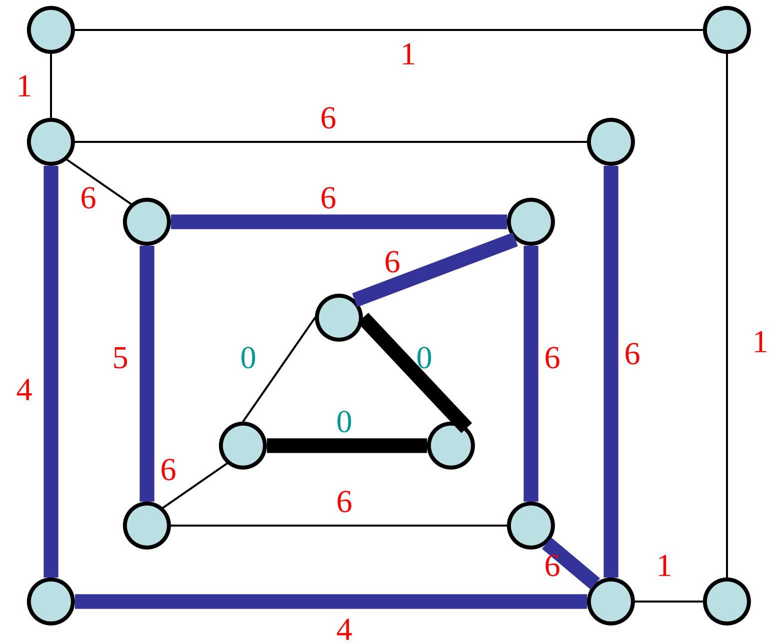
Approximation Algorithm for LTFS

[1.5, 36] [3, 36]

Define $E_i^{'} = \{e \in E(G) : l_i/(t-1) \le c(e) \le h_1\}$ $E_i^{''} = \{e \in E(G) : l_i \le c(e) \le h_1\}$

- For each connected component of $G_i^{'} = (X_i, E_i^{'})$ construct a minimum weight Steiner-tree where the terminals are vertices from $G_i^{''} = (Y_i, E_i^{''})$ and the prices are the edge weights.
- Set the price of each edge in $E_i^{''}$ to 0. The Steiner-tree edges are stored in F.



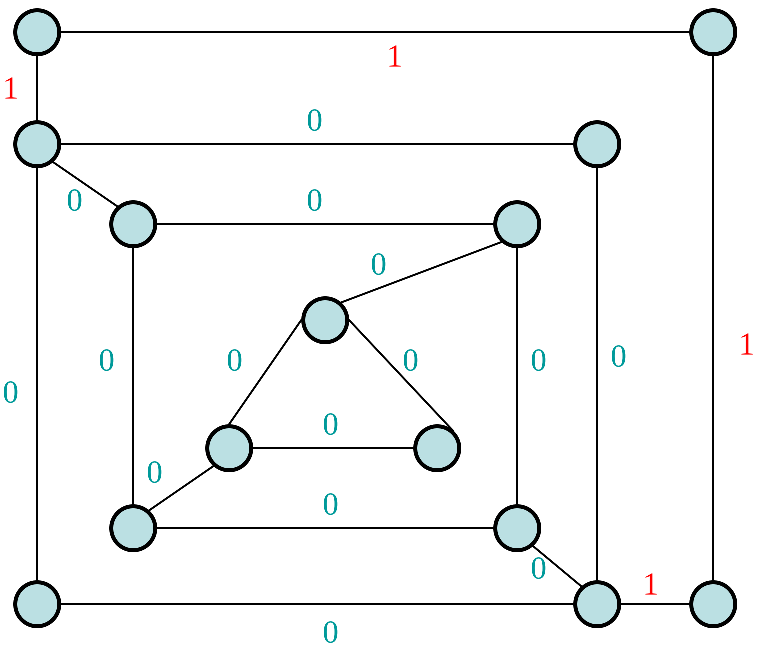


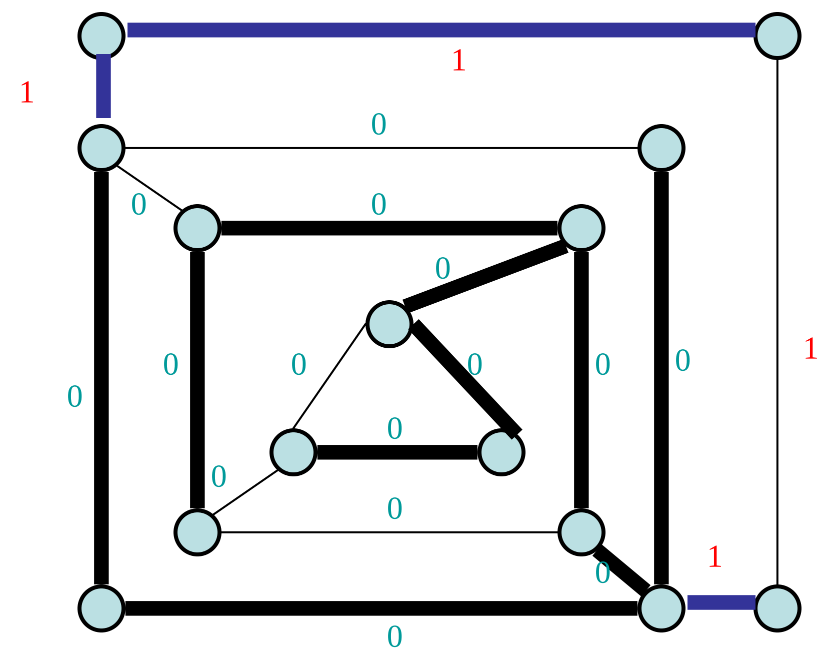
Approximation Algorithm for LTFS

[0.25, 36] **[0.5, 36]**

Define $E_i^{'} = \{e \in E(G) : l_i/(t-1) \le c(e) \le h_1\}$ $E_i^{''} = \{e \in E(G) : l_i \le c(e) \le h_1\}$

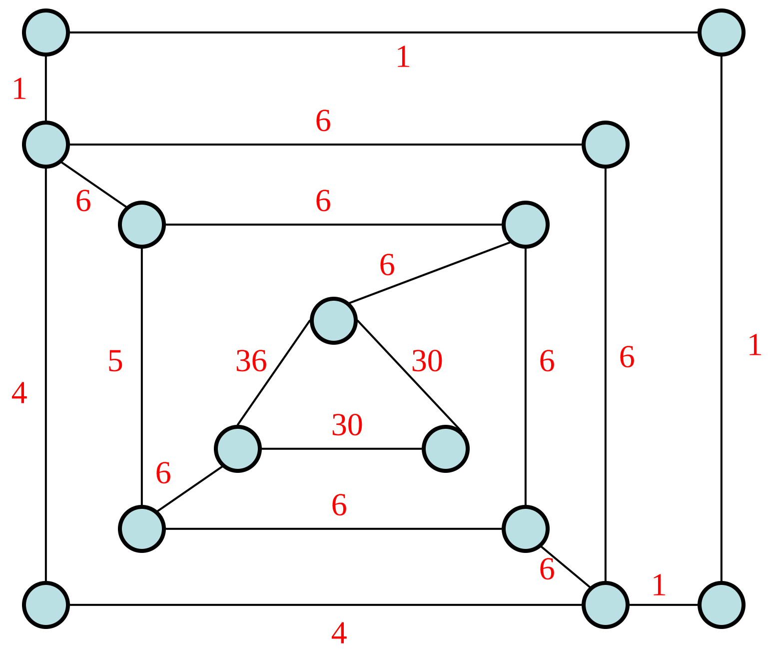
- For each connected component of $G_i^{'} = (X_i, E_i^{'})$ construct a minimum weight Steiner-tree where the terminals are vertices from $G_i^{''} = (Y_i, E_i^{''})$ and the prices are the edge weights.
- Set the price of each edge in $E_i^{''}$ to 0. The Steiner-tree edges are stored in F.



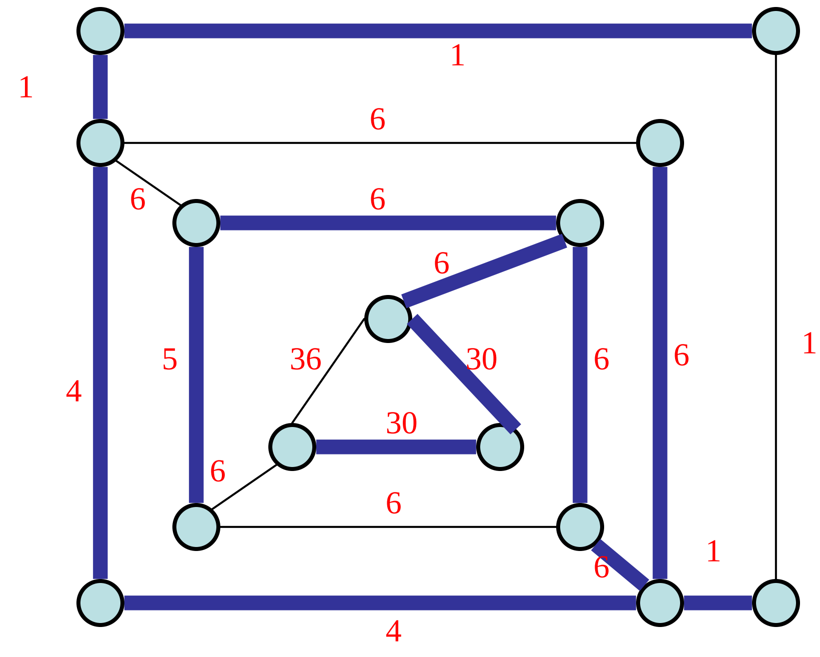


Approximation Algorithm for LTFS

Finally, construct a maximum spanning tree *T** of *H=(V,F)*, where the weight of each edge is its capacity.



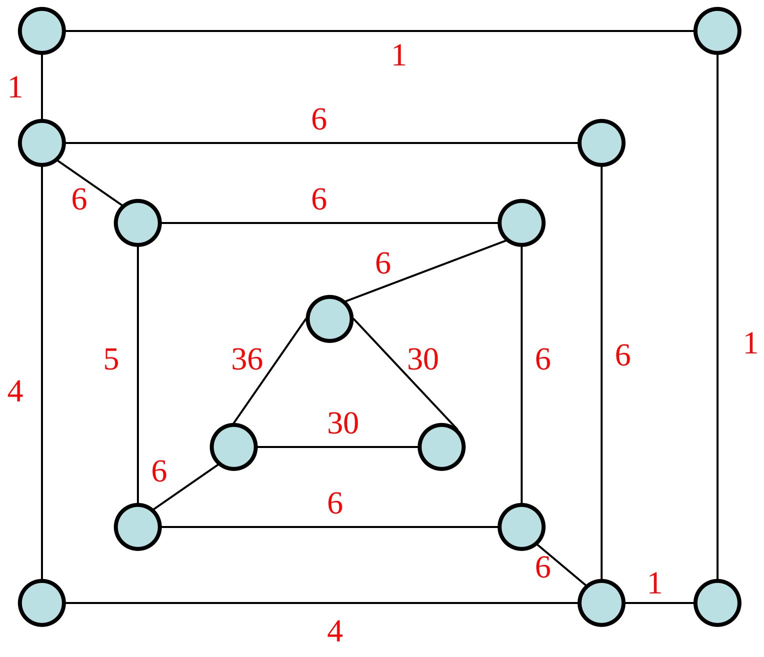
G



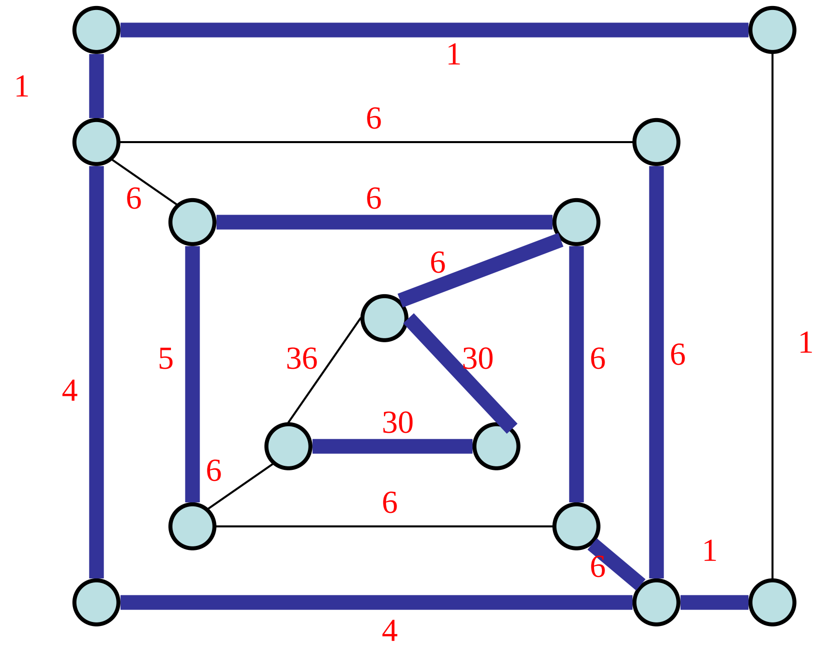
*T**

Approximation Algorithm for LTFS

Theorem 4. There exists an $(r(t\text{-}1),\ 1.55\log_r(r(t\text{-}1)))$-approximation algorithm for the ***Light Tree Flow-Spanner*** problem.



G

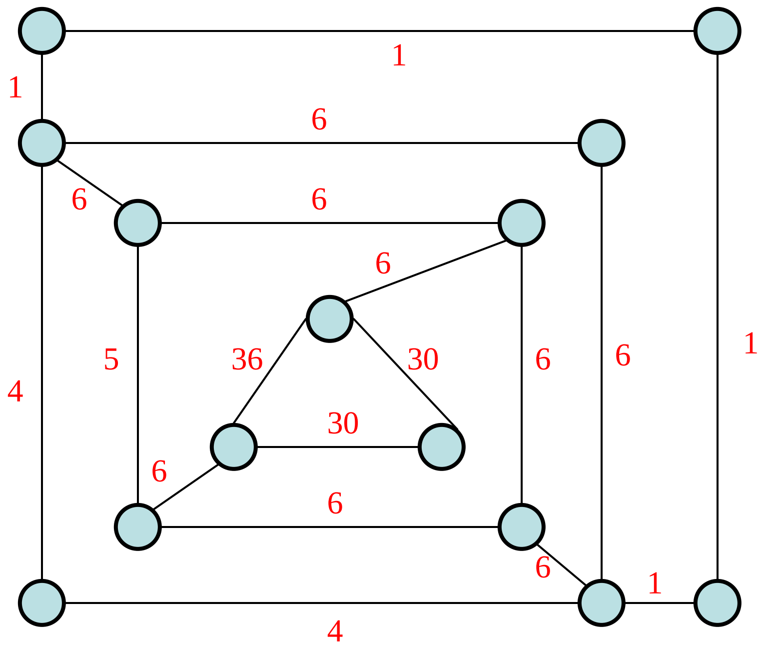


*T**

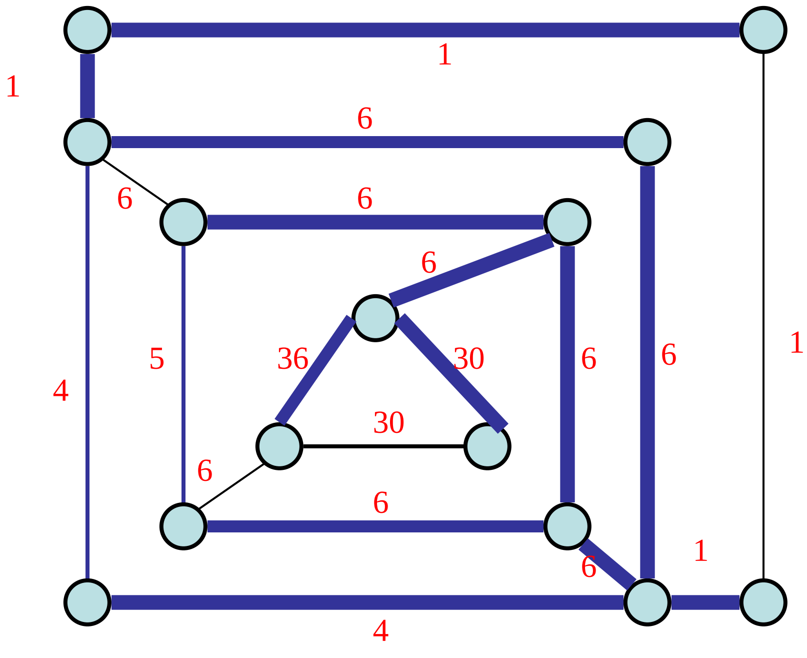
t ➔ ***r(t-1) t ,*** ***P*** ➔ ***1.55log$_r$(r(t-1)) P*** (for any r: 1<r<t)

Our Second Approximation Algorithm for LTFS

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



G



T^*

$t \rightarrow t$, $P \rightarrow (n-1)P$

Conclusion

- ***Sparse Edge-Connectivity-Spanner*** is NP-hard
  - ***Light Flow-Spanner*** is NP-hard
  - ***Sparse Flow-Spanner*** is NP-hard
  - ***Light-Edge-Connectivity-Spanner*** is NP-hard
- ***Light Tree Flow-Spanner (LTFS)*** is NP-hard
- Two approximation algorithms for ***LTFS.***

Future work

- Show that it is NP-hard even to approximate.
- Better approximations for the ***LTFS*** problem.
- Approximate solutions for the general ***LFS*** problem.

Thank You