Approximation Algorithm for Subset $k$-Connectivity

Bundit Laekhanukit
School of Computer Science, McGill University
Outline of This Talk

• Problem Formulation
• Structural Properties
• Main Algorithm
Min-cost subset $k$-connected subgraph problem (Subset $k$-Conn)
Given a graph with edge-cost, a set of terminals, and requirement $k$
We want to pay cheap cost to make a graph $k$-connected on terminals.
E.g, \((k=2)\) all terminals remain connected after removing one vertex.
Min-cost Subset $k$-Connected Subgraph Problem (subset $k$-conn)

Input:
- Graph $G = (V, E)$ with non-negative cost on edges
- A set of terminals $T$.
- An integer $k$, a requirement

Goal:
- Find a min-cost subgraph $H = (V, E')$.
- $H$ has $k$-vertex disjoint paths connecting each pair $s, t$ of terminals.

$k = 1$: Steiner tree problem $\Rightarrow$ NP-Hard
Current Status in terms of $|T|$ and $k$

UB: $|T|^2$ (trivial algorithm)  
LB: LabelCover $\approx \Omega(k^\epsilon)$

UB: $O(k^2 \log k)$-approx  
LB: APX-Hard
Our Results

$|T|^2$ (trivial algorithm)  $O(k \log^2 k)$-approx  $O(k \log k)$-approx
Given: $G = (V,E)$, edge-cost, root $r$, terminal set $T$
Goal: Find min-cost subgraph having $k$-disjoint $r,t$ path for each terminal $t$
Closely related problem rooted subset $k$-connectivity

Current Status:
UB : $O(k \log k)$-approx,  LB : $O(k^\epsilon)$-hardness [Cheriyan, L. '11]
Hardness of rooted \( k \)-conn

- Reduce Directed Steiner Forest to Rooted \( k \)-Conn on Directed Graphs
- Apply Lando-Nutov's Thm to reduce the problem to undirected graphs (with connectivity \( k'=k+|V| \))
- Hardness can be tighten to \( k^\epsilon \) by reducing it directly from LabelCover.
Reducing subset $k$-connectivity to rooted (subset) $k$-connectivity

All previous algorithms solve subset $k$-connectivity by applying rooted $k$-connectivity algorithm to $k$ terminals (pay a factor of $k$)
Comparison of Approx Ratio

Subset: $|T|^2$
Rooted: $|T|$
(trivial algorithm)

Subset: $O(k \log^2 k)$
Rooted: $O(k \log k)$

Subset & Rooted: $O(k \log k)$
Subset $k$-conn is hardest when $|T| = k$
Subset $k$-conn is hardest when $|T| = k$

$G(k)$-aprx algo for **rooted** $k$-conn

$F(k)$-aprx algo for **subset** $k$-conn for $|T| = k$
Subset $k$-conn is hardest when $|T| = k$

G($k$)-aprx algo for \textbf{rooted} $k$-conn

Choose $k$ terminals

F($k$)-aprx algo for \textbf{subset} $k$-conn for $|T| = k$
Subset $k$-conn is hardest when $|T| = k$

G($k$)-aprx algo for **rooted** $k$-conn

Choose $k$ terminals

F($k$)-aprx algo for **subset** $k$-conn for $|T| = k$
Subset $k$-conn is hardest when $|T| = k$

All terminals $k$ connected, Approx Ratio = $G(k) + F(k)$
Outline of our technique

- Use Connectivity Augmentation Framework
- Halo-set Method: Apply rooted $k$-connectivity algorithm to covering deficient sets
Connectivity Augmentation framework.
Increase connectivity from $L = 1, 2, \ldots, k$
Start from connectivity = 1
Increase connectivity to 2
Increase connectivity to 3
Pay a factor of $O(\log k)$

- LP for Conn. Aug., $L \Rightarrow (L+1)$, asks for one edge covering cut $U$
- Take integral OPT.
- Scale edges by $1/(k - L)$
  $\Rightarrow$ Sol feasible to LP
- Run $k$ times pay $O(\log k)$
Assume a graph is subset $L$-connected on $T$, and we want to increase connectivity to $L+1$
Also, our goal is to attack the case $|T| \geq 2k$
So, we assume $|T| \geq 2L$
Structure of subset $L$-connected graph
Deficient Set

- Set of vertices $U$ with $L$ neighbors, removing neighbors separates some pair of terminals.

- $|U \cap T| \leq |U^* \cap T| \implies$ call small deficient set.
Cover Deficient Set

• Add an edge between $U$ and $U^*$
Key Idea (Halo-Set method)

- Group deficient sets by notion of **halo-families**
- Goal: Pay **cheap** cost to dec #halo-fami by $\times 1/2$
Core

• An inclusionwise minimal small deficient set
Halo-family of a core $C$

- $Halo(C) = \{ U : U \text{ is a small deficient set, } U \text{ contains } C \text{ and contains no core } D \neq C \}$
Some small deficient sets are not in any halo-family

- Small deficient sets contain $\geq 2$ cores are not in any halo-family.
Halo-set Method: Framework

- **For** \( L=0,1, \ldots, k \)
  
  *connectivity augmentation*
  
  - **While** # of cores > 0
    - Compute “cores” and “halo-families”
    - Find edges covering all halo-families
  
  - **End While**
  
  *end connectivity augmentation*

- **End For**
Halo-set Method

• In each round, we cover all the halo-families.

Round 1
Halo-set Method

- Recompute and cover (new) halo-families.
Halo-set Method

- Repeat it again until no cores/halo-families left.
  \[ \Rightarrow \text{terminates in } O(\log q) \text{ round, } q = \# \text{ of cores} \]
Some difficulties

- Cores/Halo-families can intersect on terminals
- \# of Halo-families can be \( O(|T|^2) \)
- \# of Halo-families can increase (after adding edges)
- No known algo for covering all halo-families
Solve the difficulties (for $|T| \geq 2L$)

- Each terminal is in $\leq O(1)$ Halo-families
- Preprocessing $\Rightarrow$ decreases # of Halo-families to $O(L)$
- Use max # of terminals-disjoint cores as notion of progress
- **Main Algo**: Cover Halo-families by rooted $(L+1)$-conn algo.
Main Algorithm: Observation

- Running rooted \((L + 1)\)-conn algorithm with a root \(r\) covers all deficient set containing \(r\).
Main Algorithm: Observation

- If chosen root $r$ is (1) in a core $C$ or (2) in vertex-complement of $Halo(C)$, then rooted algo covers $Halo(C)$. 
Main Algorithm: Observation

- \(\text{Halo}(C)\) is **NOT covered** if chosen root \(r\) is (1) in \(\text{Halo}(C)\) but not in \(C\) or (2) is a neighbor \(\text{Halo}(C)\).
Main Algorithm: Key Lemma

For $|T| \geq 2L$, there is a terminal $r$ such that

1) There are $O(1)$ halo-families of Case 1.

2) There are $O(q/2)$ halo-families of Case 2, where $q$ is # of halo-families.

**REMARK:** Bound for (1) is $O(L / (|T| - L))$, Bound for (2) is $(q / (|T| / L))$
Main Algorithm: Description

Covering Procedure:

• Repeat
  • Find a terminal $r$ satisfying Key Lemma
  • Apply rooted $(L+1)$-conn from $r$
  • For each $Halo(C)$ of Case 1, choose any terminal $t$ in $C$ and apply rooted $(L+1)$-conn from $t$

• Until all halo-families are covered

• REMARK: We consider only the # of halo-families computed at the beginning of the round (of Halo-set Method). So, we have to repeat everything again.
Sketch of Analysis

Covering Procedure:

- **Repeat**
  - Find a terminal $r$ satisfying Key Lemma
  - Apply rooted $(L+1)$-conn from $r$
  - For each $Halo(C)$ of Case 1, choose any terminal $t$ in $C$ and apply rooted $(L+1)$-conn from $t$
- **Until** all halo-families are covered

- **Key Lemma**
  \[ \Rightarrow \text{Cover O(1) halo-fam (Case 1)} \Rightarrow \text{Remaining are } \leq q/2 \text{ halo-fam (Case 2)} \Rightarrow \text{# of halo-families decrease geometrically} \Rightarrow \text{Need O(log } q \text{) rounds} \]
Combine Everything

- Covering-Procedure solves $O(\log k)$ rooted $k$-conn
- Halo-set method terminates in $O(\log k)$ rounds
  $\Rightarrow$ Solves $O(\log^2 k)$ rooted $k$-conn for conn aug
- Pay $O(\log k)$ factor for increase subset conn to $k$.
  $\Rightarrow O(F(k)\log^3 k)$ for subset $k$-conn, where $F(k)$ is approx ratio for rooted conn augmentation.

*(There is $O(k)$-aprx algo for rooted conn aug by Nutov [FOCS'09].)*

More Analysis

$\Rightarrow$ approx ratio = $O(k \log^2 k)$ for $|T| \geq 2k$

$\quad = O(k \log k)$ for $|T| \geq k^2$
Conclusion

- We presented a LabelCover hardness for rooted subset $k$-connectivity problem.
- We showed that, for $|T| \geq 2k$, $F(k)$-approx for rooted connectivity augmentation implies $F(k) \log^3 k$ for subset $k$-connectivity.
Questions?
Thank you.