Approximation algorithm for minimum λ-edge-connected k-subgraph with metric costs

MohammadAli Safari and Mohammad R. Salavatipour

Dept. of Computing Science
University of Alberta

August, 2008

(k, λ)-subgraph problem:

- **Input:** Given a weighted undirected graph G and integer parameters k and λ
- **Output:** Find a minimum weight λ-edge-connected subgraph of G containing at least k nodes.

(k, λ)-subgraph problem:

- **Input:** Given a weighted undirected graph G and integer parameters k and λ
- **Output:** Find a minimum weight λ-edge-connected subgraph of G containing at least k nodes.

Generalizes many classical problems:

- k-MST $\equiv (k, 1)-$subgraph problem.
  Approximation factor: $\sqrt{k}$ [13], $O(\log^2 k)$ [1], $O(\log n)$ [12], constant [3, 8] and 2 [9].
- min-cost λ-edge-connected spanning graph
  $\equiv (|V(G)|, \lambda)-$subgraph problem

(k, λ)-subgraph problem in general graphs:

- Introduced recently by Lau et al. [11].
- They obtain an $O(\log^2 n)$-approximation for $(k, 2)$-subgraph problem.

(k, λ)-subgraph problem in general graphs:

- Introduced recently by Lau et al. [11].
- They obtain an $O(\log^2 n)$-approximation for $(k, 2)$-subgraph problem.
- For arbitrary λ: as hard as the k-densest subgraph problem [11]; best known approximation factor is $O(n^{\frac{1}{3}-\epsilon})$ for some $\epsilon > 0$.

(k, λ)-subgraph problem in general graphs:

- Introduced recently by Lau et al. [11].
- They obtain an $O(\log^2 n)$-approximation for $(k, 2)$-subgraph problem.
- For arbitrary λ: as hard as the k-densest subgraph problem [11]; best known approximation factor is $O(n^{\frac{1}{3}-\epsilon})$ for some $\epsilon > 0$.
- Chekuri and Korula [5]: $O(\log^2 n)$-approx for $(k, 2)$-subgraph problem with *node-connectivity constraint.*

- Here we consider the instances in which the underlying graph G is metric:
  Theorem: For the (k, λ)-subgraph problem on metric graphs, there is an $O(1)$-approximation algorithm.

- Here we consider the instances in which the underlying graph G is metric:
  Theorem: For the (k, λ)-subgraph problem on metric graphs, there is an $O(1)$-approximation algorithm.
- Note that the constant factor approximation algorithms for k-MST and k-TSP are on graphs with metric cost function.

- Here we consider the instances in which the underlying graph G is metric:
  Theorem: For the (k, λ)-subgraph problem on metric graphs, there is an $O(1)$-approximation algorithm.
- Note that the constant factor approximation algorithms for k-MST and k-TSP are on graphs with metric cost function.
- The constant in the $O(1)$ term is between 400-500.

- Here we consider the instances in which the underlying graph G is metric:
  Theorem: For the (k, λ)-subgraph problem on metric graphs, there is an $O(1)$-approximation algorithm.
- Note that the constant factor approximation algorithms for k-MST and k-TSP are on graphs with metric cost function.
- The constant in the $O(1)$ term is between 400-500.
- Our algorithm is inspired by the work of Cheriyan and Vetta [4] for subset-node-connectivity problem.

We use two basic lower bounds on the optimum solution.
Let G^* be the optimal solution and $\text{OPT} = c(G^*)$; T^* be a MST of G^*.

We use two basic lower bounds on the optimum solution.
Let G^* be the optimal solution and $\text{OPT} = c(G^*)$; T^* be a MST of G^*.

1. Minimum Spanning Tree of G^*

Using the cut-constraint in the IP-formulation of MST:
$\frac{\lambda}{2} \sum_{e \in T^*} c_e \leq \text{OPT}$

We use two basic lower bounds on the optimum solution.
Let G^* be the optimal solution and $\mathrm{OPT} = c(G^*)$; T^* be a MST of G^*.

1. Minimum Spanning Tree of G^*

Using the cut-constraint in the IP-formulation of MST:
$\frac{\lambda}{2} \sum_{e \in T^*} c_e \leq \mathrm{OPT}$

2. λ nearest neighbors

if S_u is the set of λ nearest neighbors of u and s_u is their total distance to u then $\frac{1}{2} \sum_{u \in T^*} s_u \leq \mathrm{OPT}$

We use two basic lower bounds on the optimum solution.
Let G^* be the optimal solution and $\text{OPT} = c(G^*)$; T^* be a MST of G^*.

1. Minimum Spanning Tree of G^*

Using the cut-constraint in the IP-formulation of MST:
$\frac{\lambda}{2} \sum_{e \in T^*} c_e \leq \text{OPT}$

2. λ nearest neighbors

if S_u is the set of λ nearest neighbors of u and s_u is their total distance to u then $\frac{1}{2} \sum_{u \in T^*} s_u \leq \text{OPT}$

Our algorithm presents a solution whose cost is bounded within an $O(1)$-factor of these two bounds.

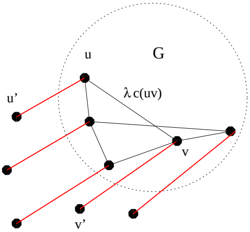
The algorithm has two main phases.

The algorithm has two main phases.

- In Phase 1, we obtain a $(k - \lambda/7, \lambda)$-subgraph, call it H, which has cost $O(\textsc{opt})$.

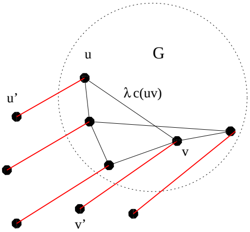
The algorithm has two main phases.

- In Phase 1, we obtain a $(k - \lambda/7, \lambda)$-subgraph, call it H, which has cost $O(\text{OPT})$.
- In Phase 2, we show how to expand H to a (k, λ)-subgraph, while keeping the cost within $O(\text{OPT})$.



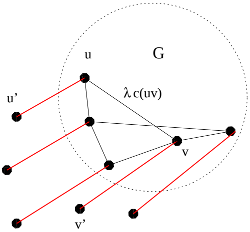
To find a low cost $(k - O(\lambda), \lambda)$-subgraph:

- Create a new graph $G'(V \cup V', E')$ from G by creating a new vertex u' for each $u \in G$ and $E' = E \cup \{uu' | u \in V\}$, $c(uu') = s_u$.



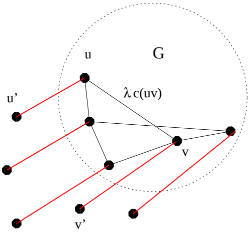
To find a low cost $(k - O(\lambda), \lambda)$-subgraph:

- Create a new graph $G'(V \cup V', E')$ from G by creating a new vertex u' for each $u \in G$ and $E' = E \cup \{uu' | u \in V\}$, $c(uu') = s_u$.
- For every other edge in G' multiply its weight by λ.



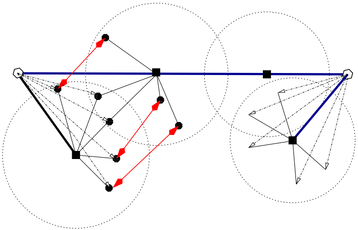
To find a low cost $(k - O(\lambda), \lambda)$-subgraph:

- Create a new graph $G'(V \cup V', E')$ from G by creating a new vertex u' for each $u \in G$ and $E' = E \cup \{uu' | u \in V\}$, $c(uu') = s_u$.
- For every other edge in G' multiply its weight by λ.
- Using a ρ-approx alg, say ST-alg, for k-Steiner tree, find a k-Steiner tree T' in G' on terminal set V'.

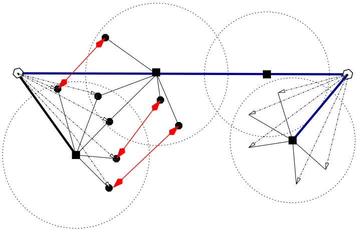


To find a low cost $(k - O(\lambda), \lambda)$-subgraph:

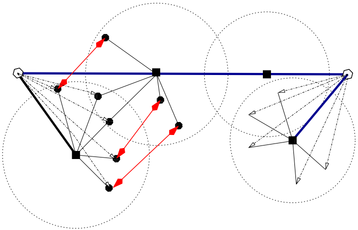
- Create a new graph $G'(V \cup V', E')$ from G by creating a new vertex u' for each $u \in G$ and $E' = E \cup \{uu' | u \in V\}$, $c(uu') = s_u$.
- For every other edge in G' multiply its weight by λ.
- Using a ρ-approx alg, say ST-alg, for k-Steiner tree, find a k-Steiner tree T' in G' on terminal set V'.
- Note that T' minimizes (approximately) $\sum_{u \in T'} s_u + \lambda \sum_{e \in T'} c_e$, and has at least k nodes of G.



- Using the two lower bounds mentioned, it is easy to show that:
  Claim:$c(T') = 4\rho\mathrm{OPT}$.
  Let $T_0 \subseteq G$ be the tree obtained by deleting dummy vertices of T'.



- Using the two lower bounds mentioned, it is easy to show that: Claim: $c(T') = 4\rho\mathrm{OPT}$.
  Let $T_0 \subseteq G$ be the tree obtained by deleting dummy vertices of T'.
- We'll pick some vertices $v \in T_0$ such that $|\bigcup_v s_v| \approx k$.
  Make a clique of each such vertex v together with its neighbors in S_v.



- Using the two lower bounds mentioned, it is easy to show that: Claim:$c(T') = 4\rho\mathrm{OPT}$.
  Let $T_0 \subseteq G$ be the tree obtained by deleting dummy vertices of T'.
- We'll pick some vertices $v \in T_0$ such that $|\bigcup_v s_v| \approx k$. Make a clique of each such vertex v together with its neighbors in S_v.
- For every $(u, v) \in T_0$ put a matching between the vertices in $S_u - S_v$ and $S_v - S_u$ to obtain λ-edge-connectivity.

More details on how to do it...

- Let $v_1, \ldots, v_k$ be an ordering of vertices of T_0 s.t. $s_{v_1} \leq s_{v_2} \leq \ldots s_{v_k}$.

More details on how to do it...

- Let $v_1, \ldots, v_k$ be an ordering of vertices of T_0 s.t. $s_{v_1} \leq s_{v_2} \leq \ldots s_{v_k}$.
- We call the set S_{v_i} the ball with center v_i and $B_{v_i} \subseteq S_{v_i}$, called the core, is the set of nodes with distance at most $2s_{v_i}/\lambda$ to v_i.

More details on how to do it...

- Let $v_1, \dots, v_k$ be an ordering of vertices of T_0 s.t. $s_{v_1} \leq s_{v_2} \leq \dots s_{v_k}$.
- We call the set S_{v_i} the ball with center v_i and $B_{v_i} \subseteq S_{v_i}$, called the core, is the set of nodes with distance at most $2s_{v_i}/\lambda$ to v_i.
- We use a clustering to obtain a set of Active/Inactive balls:
  - Active vs. Inactive Balls: Every vertex is active unless it is close to an active ball with smaller s_u value.
  - The cores of active balls are disjoint.

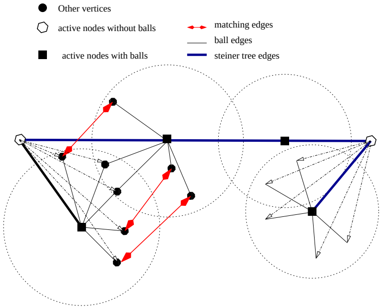
More details on how to do it...

- Let $v_1, \ldots, v_k$ be an ordering of vertices of T_0 s.t. $s_{v_1} \leq s_{v_2} \leq \ldots s_{v_k}$.
- We call the set S_{v_i} the ball with center v_i and $B_{v_i} \subseteq S_{v_i}$, called the core, is the set of nodes with distance at most $2s_{v_i}/\lambda$ to v_i.
- We use a clustering to obtain a set of Active/Inactive balls:
  - Active vs. Inactive Balls: Every vertex is active unless it is close to an active ball with smaller s_u value.
  - The cores of active balls are disjoint.
- Let i^* be the smallest index such that $U_{i*} = \bigcup_{\text{active } v_j,\, j \leq i*} S_{v_j}$ has at least $k - \lambda/7$ nodes. We discard vertices v_j with $j > i^*$.

More details on how to do it...

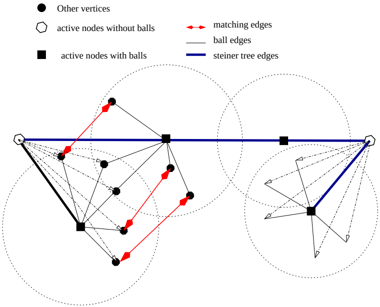
- Let $v_1, \ldots, v_k$ be an ordering of vertices of T_0 s.t. $s_{v_1} \leq s_{v_2} \leq \ldots s_{v_k}$.
- We call the set S_{v_i} the ball with center v_i and $B_{v_i} \subseteq S_{v_i}$, called the core, is the set of nodes with distance at most $2s_{v_i}/\lambda$ to v_i.
- We use a clustering to obtain a set of Active/Inactive balls:
  - Active vs. Inactive Balls: Every vertex is active unless it is close to an active ball with smaller s_u value.
  - The cores of active balls are disjoint.
- Let i^* be the smallest index such that $U_{i^*} = \bigcup_{\text{active } v_j,\, j \leq i*} S_{v_j}$ has at least $k - \lambda/7$ nodes. We discard vertices v_j with $j > i^*$. Note that $k - \frac{\lambda}{7} \leq |U_{i^*}| \leq k + \frac{6\lambda}{7}$.

Algorithm for Phase 1



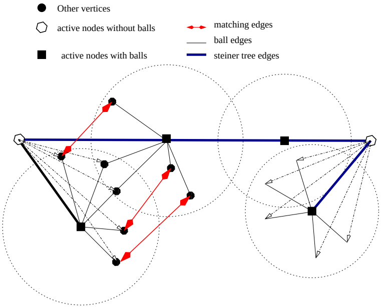
1. By short-cutting over non-active nodes in T_0, we obtain tree T_1. Then for each active nodes $v_j \in U_{i*}$ make a clique on S_{v_j}.

Algorithm for Phase 1



1. By short-cutting over non-active nodes in T_0, we obtain tree T_1. Then for each active nodes $v_j \in U_{i*}$ make a clique on S_{v_j}.
2. For every $(u, v) \in T_1$ put a matching between the vertices in $S_u - S_v$ and $S_v - S_u$ to obtain λ-edge-connectivity.

Algorithm for Phase 1



1. By short-cutting over non-active nodes in T_0, we obtain tree T_1. Then for each active nodes $v_j \in U_{i*}$ make a clique on S_{v_j}.
2. For every $(u, v) \in T_1$ put a matching between the vertices in $S_u - S_v$ and $S_v - S_u$ to obtain λ-edge-connectivity.
3. It can be shown that the resulting graph H is λ-edge-connected, has at least $k - \lambda/7$ nodes and has cost at most $28\rho\text{OPT}$.

Phase 2:

How to expand H to have size k.

$\forall u \in G \backslash H$: let $d(u, H)$ be the distance between u and H.

Phase 2:

How to expand H to have size k.
$\forall u \in G \backslash H$: let $d(u, H)$ be the distance between u and H.

Case 1

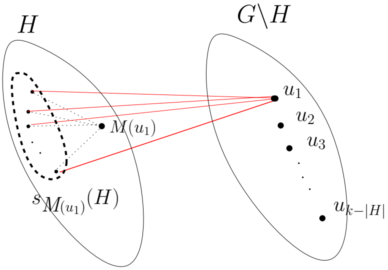
If there is a set $A \subseteq G \backslash H$ of $k - |H|$ vertices s.t. has a low-cost matching between A and H then we can augment H with small cost.

Phase 2:

How to expand H to have size k.
$\forall u \in G \backslash H$: let $d(u, H)$ be the distance between u and H.

Case 1

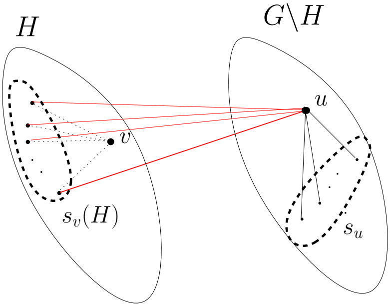
If there is a set $A \subseteq G \backslash H$ of $k - |H|$ vertices s.t. has a low-cost matching between A and H then we can augment H with small cost.



- connect each u_i to $S_{M(u_i)}$ to obtain λ-edge-connectivity. Total cost added: $\lambda c(M) + 2c(H)$
- Show if $|G^* \backslash H| \leq \lambda/3$, then this can be done with $c(M) \leq \frac{6\text{OPT}}{\lambda}$

Case 2

If there is a vertex $u \in G \backslash H$ s.t. $s_u + d(u, H)$ is small and S_u contains at least $\lambda/7$ vertices in $G \backslash H$ then we can augment H with small cost.



- It can be shown that if $|G^* \backslash H| > \lambda/3$, i.e. Case 1 does not happen, then this happens
- The cost of augmenting H in this case is $\leq 12\text{OPT} + 3c(H)$.

Conclusion

- So we can extend H to a (k, λ)-subgraph by spending a total of at most $12\text{OPT} + 3c(H)$.
- Recalling that $c(H) = O(\text{OPT})$, the total approximation ratio is $18 + 108\rho$ with $\rho \leq 4$ being the ratio for k-Steiner tree.
- Getting a small constant factor approximation seems challenging, for general values of λ.
- For general cost functions, even for the special case of $\lambda = 3$, there is no known non-trivial approximation algorithm or lower bound.

- B. Awerbuch, Y. Azar, A. Blum and S. Vempala, *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Computing 28(1):254-262, 1999.
- A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff, *Approximation Algorithms for Orienteering and Discounted-Reward TSP*, SIAM J. on Computing 28(1):254-262, 1999. Earlier version in Proc of STOC 1995.
- A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k-MST problem*, J. Comput. Syst. Sci. 58(1): 101-108, 1999. Earlier in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96), pp. 442-448.
- J. Cheriyan and A. Vetta, *Approximation algorithms for network design with metric costs*, In Proceedings of the

thirty-seventh annual ACM symposium on Theory of computing (STOC) 2005, 167–175.

- C. Chekuri and N. Korula, *Min-Cost 2-Connected Subgraphs with k Terminals*, manuscript 2008, available at http://arxiv.org/abs/0802.2528.
- C. Chekuri, N. Korula, and M. Pál, *Improved Algorithms for Orienteering and Related Problems*, In Proc of ACM-SIAM SODA, 2008.
- F. Chudak, T. Roughgarden, and D. P. Williamson, *Approximate MSTs and Steiner trees via the primal-dual method and Lagrangean relaxation*, Math. Program. 100(2):411-421, 2004.
- N. Garg, *A 3-Approximation for the minim tree spanning k vertices*, In Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS), 302-309, 1996.

N. Garg, *Saving an epsilon: a 2-approximation for the k-MST problem in graphs*, In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC), 396 - 402, 2005.

K. Jain, *A factor* 2 *approximation algorithm for the generalized Steiner network problem*, Combinatorica, 21:39-60, 2001.

L. Lau, S. Naor, M. Salavatipour, and M. Singh, *Survivable Network Design with Degree or Order Constraints*, To appear in SIAM J. on Computing. Earlier version in Proceedings of the thirty-nineth annual ACM symposium on Theory of computing (STOC), 2007.

S. Rajagopalan and V. Vazirani, *Logarithmic approximation of minimum weight k trees*, unpublished manuscript, 1995.

R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrants, and S.S. Ravi, *Spanning trees short or small*, SIAM Journal on Discrete Mathematics, 9(2):178-200, 1996.

A. Schrijver, Combinatorial Optimization, Springer, 2003.