A Greedy Approximation Algorithm for the Group Steiner Problem

Chandra Chekuri, Guy Even, Guy Kortsarz

Presenter: Ricardo A. Collado

Rutgers University

Network Design and Game Theory

Application History

Single-Port Wire Routing in VLSI

- Each terminal is a single port
- Looks for a minimum length net containing all the terminals
- Application of the classical Steiner tree problem

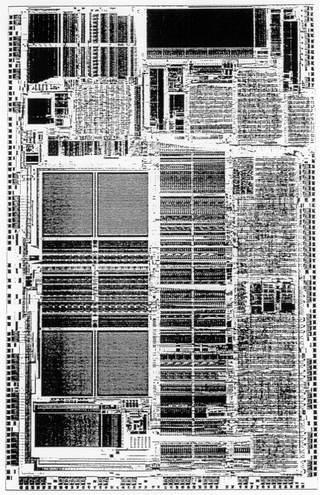
Multi-Port Wire Routing in VLSI

- Each terminal is assigned a collection (group) of ports
- Looks for a minimum length net containing at least one port from each terminal group
- Application of the group Stiner problem

Multi-Port Routing

Advantages

- Flexibility in placement of terminals
- Examines different choices of module placement and orientation
- More interaction between placement and routing phases
- Allows better optimizations in the design



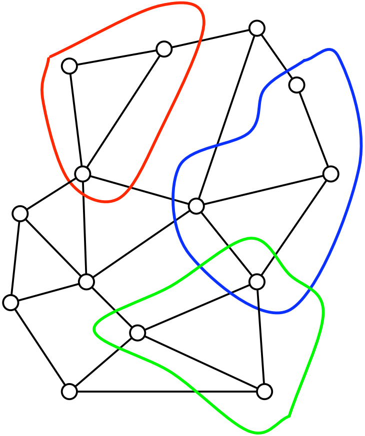
Group Steiner Problem Definition

Input:

- Undirected edge-weighted graph $G = (V, E, w)$
- Groups of vertices $g_1, \ldots, g_m \subset V$

Objective:

Find a minimum weight tree that contains at least one vertex from each group



$$w(e) = 1, \forall e \in E$$

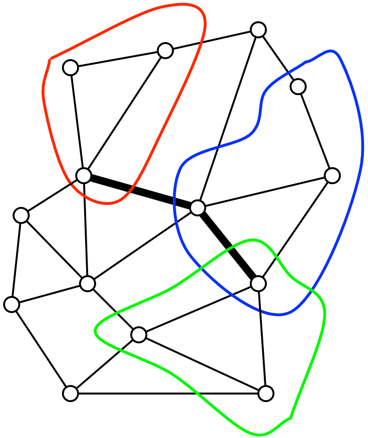
Group Steiner Problem Definition

Input:

- Undirected edge-weighted graph $G = (V, E, w)$
- Groups of vertices $g_1, \ldots, g_m \subset V$

Objective:

Find a minimum weight tree that contains at least one vertex from each group

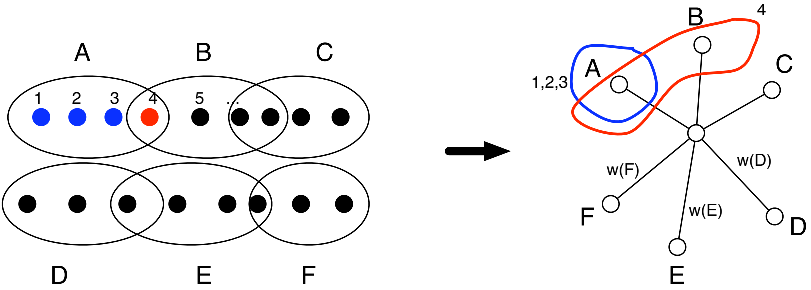


$$w(e) = 1, \forall e \in E$$

Complexity

- Direct generalization of **set cover**:
- Cannot be approximated to a factor $o(\ln k)$ unless $P = NP$ (even if G is a star)

Reduction:



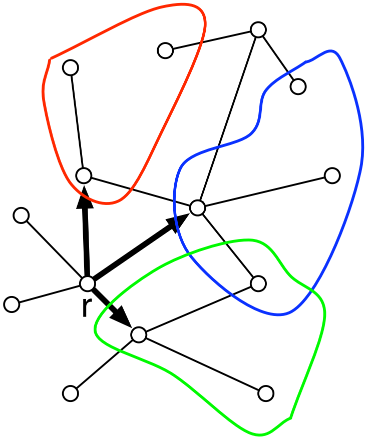
Group Steiner Problem on Trees:

Input:

- Rooted edge-weighted tree $G = (V, E, r, w)$
- Groups of vertices $g_1, \ldots, g_m \subset V$

Objective:

Find a minimum r-rooted tree that contains at least one vertex from each group



$w(e) = 1, \forall e \in E$

Outline

We will show a combinatorial poly-logarithmic approximation to the group Steiner problem on trees.

- Previous algorithms were based on solving an LP relaxation problem
- Approximation of $O(\frac{1}{\epsilon} \cdot (\log n)^{1+\epsilon} \cdot \log m)$
- Sightly worse than previous ratio $O(\log n \log m)$
- We extend probabilistically to problem on general graphs by using tree metrics

Two Main Ideas

Preprocessing:

- Height reducing transformation:
  Reduces the hight of the tree to $O(\log_\alpha n)$ with a factor of $O(\alpha)$ in weight of the optimal solution
- Degree reducing transformation:
  Reduces the max degree to $\beta + 1$ while increasing height by $O(\log_{\beta/2} n)$ and not increasing the weight of optimal

Geometric Greedy Algorithm:

- Reduce the number of recursive calls by geometric search
- Avoids sub-trees that cover few groups

Trivial Preprocessing

- Eliminate every nonterminal leaf
- Eliminate every nonterminal interior node of degree two
  - The number of nodes is $O(n)$
- Add a new group containing all nodes
- Scale edge weights such that $w(e) > 0$ implies that $w(e) > 1$

Definitions and Notation

Let T' be an r-rooted subtree of T

- $n(T') =$ Number of terminals in T'
- $m(T') =$ Number of groups in T'
- $w(T') =$ Weight of T'
- $h(T') =$ Height of T'
- Density of T': $\gamma(T') = \frac{w(T')}{m(T')}$
- A *z-cover* is a set $S \subseteq V(T)$ that covers z terminals

Faithful Trees

Definition

The tree B is an α-faithful representation of the tree A if there is $\pi : V(A) \to V(B)$ such that:

1. For every $S \subseteq A$, $w_B\left(B[\pi(S)]\right) \leq \alpha \cdot w_A\left(A[S]\right)$
2. For every $S' \subseteq \pi\left(V(A)\right)$, $w_A\left(A[\pi^{-1}(S')]\right) \leq w_B\left(B[S']\right)$

Theorem

Let B be an α-faithful representation of A. A β-approximate z-cover in B induces an $(\alpha \cdot \beta)$-approximate z-cover in A.

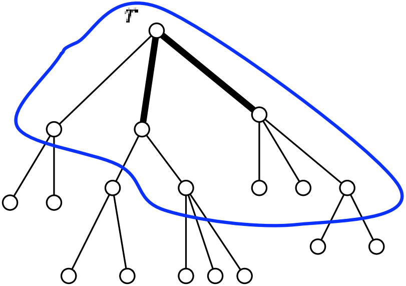
Definition

- A node u is α-*light* with respect to T_r if $n_u \leq n_r/\alpha$
- A node u is α-*heavy* with respect to T_r if $n_u > n_r/\alpha$
- A subtree $Q \subseteq T_r$ is an α-*decomposition* of T_r if $r \in Q$ and every leaf of Q is maximally α-light
- The *skeleton* of an α-decomposition Q is the subtree $sk(Q) \subset Q$ induced by all the *alpha*-heavy nodes in Q

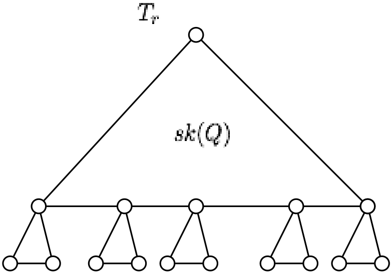
Remark:

- We can obtain an α-decomposition by exploring T_r with depth first search twice
- Every leaf in $sk(Q)$ is minimally α-heavy
- The number of leaves in $sk(Q)$ is at most α

Example



$n = 11$, $\alpha = 3$, and $\frac{n}{\alpha} \simeq 3.7$



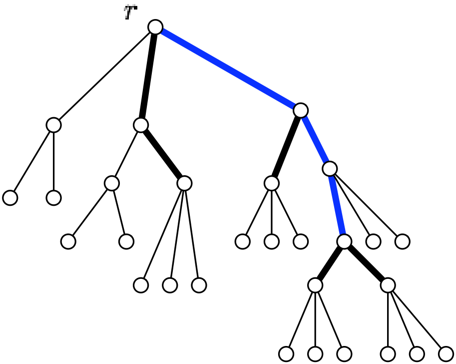
Global structure of an α-decomposition

Branches

Definition

A *branch* is a maximal subpath in $sk(Q)$ between two branching points

Example:
$n = 18$, $\alpha = 9$, and $\frac{n}{\alpha} \simeq 2$



Bunches

Definition

Fix a branch B of $sk(Q)$. Denote the endpoint of B closer to r by v. Form the following *bunches*:

$$B_0 = \{u \in B \,|\, w(path(v,u)) = 0\}$$
$$B_i = \left\{u \in B \,\middle|\, w(path(v,u)) = \in [2^{i-1}, 2^i)\right\}$$

Recall that nonzero edge weights are at least 1, so there are no vertices between B_0 and B_1

For every branch B, the following subtree is constructed. Let r' denote the root of Q'. Add a node $v(B)$ in Q', that corresponds to v, and an edge $(r', v(B))$. The edge $(r', v(B))$ is given weight equal to the weight of the path from r to v. The bunches B_i are promoted as follows. For every non-empty bunch B_i, add a new node b_i and an edge $(v(B), b_i)$. For every leaf $\ell \in \mathcal{L}(Q)$ hanging from a node in B_i, we create a leaf $\ell' \in \mathcal{L}(Q')$ that hangs from b_i. Weights are assigned as follows: (a) $w(v(B), b_0) = 0$, (b) $w(v(B), b_i) \leftarrow 2^i$, if $i > 0$, and (c) $w(b_i, \ell') \leftarrow w(p(\ell), \ell)$, for a leaf ℓ hanging from a vertex in B_i.

The mapping π maps the nodes $V(Q)$ to $V(Q')$ as follows. The root of Q is mapped to the root of Q'. For a branch B, all the nodes in B_i are mapped to the node b_i. Every leaf $\ell \in \mathcal{L}(Q)$ is mapped to its counterpart $\ell' \in \mathcal{L}(Q)$.

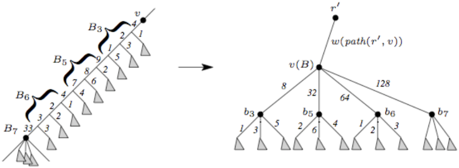


Figure 1: Promotion of bunches along a single branch. Depth of light leaves after promotion is three.

Height Reducing Transformation

Definition

Height reducing transformation:

- If r is a leaf, return a copy of T_r
- Otherwise
  - An α-decomposition Q is computed
  - A height 3 subtree Q' is created from Q
- Recurse: the mapping π is defined in every step of the recursion as described before

Since promotion substitutes Q by Q' of height 3 we get the recurrence $h(n) \leq 3 + h(n/\alpha)$. Thus $h(T') \leq 3 \log_\alpha n$.

Analysis of HR-Transformation I

Theorem

T' is an $O(\alpha)$-faithful representation of T

Let $Q_1, \ldots, Q_k$ be the sequence of α-decompositions computed during the HR-transformation. Let Q_i' be the height-3 subtree of T' used to promote Q_i. Clearly $\{Q_i\}_i, \{Q_i'\}_i$ partition $E(T), E(T')$, respectively.

Let $S \subseteq V(T)$, we assume that S contains all the *border* points (points in at least two Q_i's) in $T[S]$. Then

$$w_T(T[S]) = \sum_{i=1}^{k} w_{Q_i}(Q_i[S_i])$$

$$w_{T'}(T'[S']) = \sum_{i=1}^{k} w_{Q_i'}(Q_i'[S_i'])$$

where $S_i = S \cap V(Q_i)$, $S' = \pi(S)$, and $S_i' = \pi(S_i)$.

Analysis of HR-Transformation II

The previous hold if $S' \subseteq \pi(V(T))$ and $S = \pi^{-1}(S')$, hence it suffices to show that Q_i' is an $O(\alpha)$-faithful representation of Q_i, $1 \leq i \leq k$.

Main Idea:

There is a separate subtree in Q' for every branch in Q and this means that $w(e)$ is counted multiple times (once pre branch below e). But the number of branches is $O(\alpha)$, so the increase in weight can be bounded by $O(\alpha)$.

To bound the multiple counting within each branch we use the fact that the weights of edges $(v(B), b_i)$ increase exponentially with i, and hence their sum is dominated by the heaviest edge.

Let $\beta > 0$

- If u is a leaf, return u
- Otherwise, let $v_1, \ldots, v_k$ be the children of u. Do:
  - The β-heavy v_i's are not changed: the edges (u, v_i) are kept and their weights are not modified
  - The β-light children of u are grouped into minimal β-heavy bunches
  - For every new bunch B, a new node b is created, an edge (u, b) is added as well as edges between b and the children of u in B
  - The new edge weight is: $w(u, b) = 0$ and $w(b, v_i) = w(u, v_i)$
- Recurse

Important Results

Claim 2.4 *Let $\alpha > 1$. There exists a linear time algorithm that, given a rooted tree T with n nodes, computes an $O(\alpha)$-faithful representation T' of T such that $h(T') = O(\log_\alpha n)$.*

Claim 2.5 *Let $\beta \geq 3$. There exists a linear time algorithm that, given a rooted tree T with n nodes, computes a 1-faithful representation T' of T such that: $h(T') \leq h(T) + \lfloor \log_{\beta/2} n \rfloor$ and every node in T' has at most β children.*

By choosing *alpha* $= \log^\epsilon n$ and $\beta = \log n$ we obtain a tree with height $O(\frac{1}{\epsilon} \log n / \log \log n)$ and maximum degree $O(\log n)$. Further, we are guaranteed that there is a solution in this tree of weight at most $O(\log^\epsilon n)$ times the weight of an optimal solution in the input tree.

Algorithm 2 Modified-GS($T_{r'}, z'$) - Modified GS Algorithm (uses geometric search).

stopping condition: if r' is a leaf **then** return ($T_{r'}$).
Initialize: $cover \leftarrow \emptyset$, $z^{res} \leftarrow z'$, and $T^{res} \leftarrow T_{r'}$.
while $z^{res} > 0$ **do**
  recurse: for every $u \in \text{children}(r')$ and
  for every z'' power of $(1+\lambda)$ in $\left[\frac{1}{\deg(r')\cdot(1+\frac{1}{\lambda})\cdot(1+\lambda)} \cdot z^{res}, z^{res}\right]$

$$C_{u,z''} \leftarrow \text{Modified-GS}(T_u, z'').$$

  select: (pick the lowest density tree)

$$T_{aug} \leftarrow \text{MIN-DENSITY}\left\{C_{u,z''} \cup \{(r', u)\}\right\}.$$

  update:

  (a) $cover \leftarrow cover \cup T_{aug}$.

  (b) $z^{res} \leftarrow z' - m(cover)$.

  (c) remove all groups covered by T_{aug} from T^{res}.

  (d) **if** first time $m(cover) \geq z'/h(T_{r'})$ **then** $cover_h \leftarrow cover$.

end while
return (lowest density tree $\in \{cover, cover_h\}$).

Two New Ideas

1. In Line 4: Small subtrees are avoided in the sense that the demand value is at least

$$\frac{1}{deg(r') \cdot (1 + \frac{1}{\lambda}) \cdot (1 + \lambda)} \cdot z^{res}$$

2. The algorithm stores the first partial cover that covers at least $z'/h(T_{r'})$ groups (this is used during the proofs)

Theorem

Let Δ be the max deg of $T_{r'}$ and let $\beta = \Delta(1 + 1/\lambda)(1 + \lambda)$. The running time of Modified-GS$(T_{r'}, z')$ is $O\left(n\alpha^{h(T_{r'})}\right)$ where $\alpha = \beta \cdot h(T_{r'}) \cdot \log z' \cdot \Delta \cdot \log_{1+\lambda} \beta$. If $h(T_{r'}) = O(\log n / \log\log n)$, $\Delta = O(\log n)$ and $1 \leq 1/\lambda = O(\log n)$, then the running time is polynomial in n and m.

Theorem

$$\gamma(T_{aug}) \leq (1 + \lambda)^{2h(T_{r'})} \cdot h(T_{r'}) \cdot \gamma(OPT(T_{r'}^{res}, z^{res})).$$

Theorem 3.5 *Let I be an instance of the group Steiner problem on a tree T of height $O(\log n/\log\log n)$ and maximum degree $O(\log n)$. Then* Modified-GS *runs in time polynomial n and m and gives an $O(h(T)\log m)$-approximation.*

Proof: Choose $\lambda = 1/h(T)$ in Modified-GS. For this choice of λ and the bounds on the height and degree of T it follows from Lemma 3.3 that Modified-GS runs in time polynomial in n and m.

From Lemma 3.4, we obtain that $\gamma(T_{aug}) \leq (1+1/h(T))^{2h(T)} \cdot h(T) \cdot \gamma(\textsc{opt}(T^{res}, z^{res}))$. Therefore $\gamma(T_{aug}) \leq e^2 h(T)\gamma(\textsc{opt}(T^{res}, z^{res}))$. It follows that we obtain an $O(h(T)\log m)$ approximation. □

Corollary 3.6 *For any fixed $\varepsilon > 0$, there is a polynomial time recursive greedy algorithm for the group Steiner problem on trees with an approximation ratio of $O(\frac{1}{\varepsilon} \cdot (\log n)^{1+\varepsilon} \cdot \log m)$.*

Proof: Use Claim 2.4 with $\alpha = \log^{\varepsilon} n$ to reduce the height of the input tree to $O(\log n/\log\log n)$ and use Claim 2.5 with $\beta = \log n$ to reduce the maximum degree of the tree to $O(\log n)$ while still keeping the height $O(\log n/\log\log n)$. These transformations worsen the approximation ratio by a multiplicative factor of $O(\log^{\varepsilon} n)$. Applying the algorithm Modified-GS to the transformed tree gives the desired result. □