[theorem]Lemma

Optimal Hierarchical Decompositions for Congestion Minimization in Networks

A. Giannakopoulos

28 Ιουνίου 2013

Decomposition Tree

A decomposition tree for the graph G is a rooted tree $T = (V_t, E_t)$ whose leaf nodes correspond to nodes in G, i.e. there is a one-to-one relation between nodes in G and leaf nodes in T.

More formally there is a node mapping function $m_V : V_t \rightarrow V$ with a bijection between leaf nodes on T and nodes in G.

We are also given a function $m_E : E_t \rightarrow E^*$. that maps an edge $e_t = (u_t, v_t)$ of T to a path P_{uv} with $u = m_V(u_t)$ and $v = m_V(v_t)$ in G.

At last we have functions $m_V^{'} : V \rightarrow V_t$ and $m_E^{'} : E \rightarrow E_t^*$.

Multicommodity Flow

A multicommodity flow in a graph with n nodes is given by $\binom{n}{2}$ flows one flow for each unordered pair u, v.
When we consider a multicommodity flow on a decomposition tree we assume that the commodities are only formed by pairs of leaf nodes. For a multicommodity flow f_T on a decomposition tree we use $m(f_T)$ to denote the multicommodity flow that is obtained by mapping f_T to G via the edge-mapping function m_E.
Formally, if an edge carries flow $f_T^i(e_t)$ for commodity i on a tree edge e_t, then the graph edge e carries flow $\sum_{e_t \in E_t : e \in m_E(e_t)} f_T^i(e_t)$ for commodity i.

The Minimum Communication Cost Tree Problem.

In the MCCT-problem we are given an undirected graph $G = (V, E)$. Every edge $e \in E$ has an associated length $l(e)$, and we use d_{uv} to denote the shortest path between nodes u, v.
Further more we are given a function $r : V \times V \rightarrow \mathbb{R}_0^+$ that specifies an amount of traffic that has to be sent between u and v.
The goal is to route the requirements in a tree-like fashion while minimizing the total cost. Formally, the task is to construct a decomposition tree $T = (V_t, E_t)$ that minimizes

$$cost(T) = \sum_{(u,v)} d_T(u, v) \cdot r(u, v).$$

where $d_T(u, v)$ denotes the distance when connecting u, v.

The Minimum Communication Cost Tree Problem.

We can write the above cost in a different way. A tree edge $e_t = (u_t, v_t)$ partitions the leaf nodes of the tree and, hence, the nodes of the graph, into two disjoint sets V_{u_t} and V_{v_t}.
Let $r(e_t) := \sum_{u \in V_{u_t}, v \in V_{v_t}} r(u, v)$ denote the total requirement that has to cross the corresponding cut. All this traffic has to be forwarded via the path $m_E(e_t)$.
We define the load $load_T(e)$ that is induced on a edge $e \in E$ by tree T as

$$load_T(e) := \sum_{e_t \in E_t : e \in m_E(e_t)} r(e_t)$$

which is the total traffic that goes over e.
With these definitions we can write the cost of a decomposition tree for MCCT instance as

$$cost(T) = \sum_{e \in E} load_T(e) \cdot l(e).$$

Fakcharoenphol, Rao and Talwar Theorem.

Theorem (Fakch,Rao,Talwan)

Given an instance for MCCT problem, a solution with cost $O(\log n) \cdot \sum_e r(e) \cdot l(e)$ can be computed in polynomial time.

Approximating the Bottlenecks Of a Graph By a Tree.

In this problem we are given a graph $G = (V, E)$ together with a bandwidth function $c : V \times V \to \mathbb{R}^+$ that describes the bandwidth of the edges in E with the convention $c(u, v) = 0$ iff $(u, v) \notin E$.
Given a decomposition tree we define the capacity $c(u_t, v_t)$ of a tree edge as

$$c(u_t, v_t) := \sum_{u \in V_{u_t}, v \in V_{v_t}} c(u, v),$$

Given a multicommodity flow in G we want to compare its congestion in G, to its congestion in T.

Approximating the Bottlenecks Of a Graph By a Tree.

Theorem

Suppose you are given a multicommodity flow f in G with congestion C_G. Then the flow $m^{'}(f)$ obtained by mapping f to some decomposition tree T results in a flow in T that has congestion $C_T \leq C_G$.

Απόδειξη.

Suppose an edge $e_t = (u_t, v_t)$ in the tree has congestion C_T All traffic that traverses this edge in T has to traverse the cut in G between V_{u_t} and V_{v_t}. The total capacity of all edges over this cut is exactly equal to $c(e_t)$. Hence, one of these edges must have relative load at least C_T.This gives $C_T \leq C_G$. □

Approximating the Bottlenecks Of a Graph By a Tree.

We define the load of an edge e as

$$load_T(e) := \sum_{e_t \in E_t : e \in m_E(e_t)} c(e_t).$$

We also define the relative load of an edge e as

$$rload_T(e) := \frac{load_T(e)}{c(e)}$$

We are looking for a convex combination of decomposition trees such that for every edge the expected relative load is small, i.e.

$$minimize \beta := \max_{e \in E} \{ \sum_i \lambda_i rload_{T_i}(e) \}.$$

Approximating the Bottlenecks Of a Graph By a Tree.

Lemma

Suppose we are given a convex combination of decomposition trees with maximum expected relative load β, and suppose that we are given for each tree T_i a multicommodity flow f_i that has congestion 1 in T_i. Then, the multicommodity flow $\sum_i \lambda_i m_{T_i}(f_i)$ has congestion at most β when mapped to G.

Απόδειξη.

Fix a tree T_i. Routing the flow f_i in the tree generates congestion at most 1, which means that the amount of traffic that is send along a tree edge $e_t = (u_t, v_t)$ is at most $c(e_t)$. Hence, the total traffic that is induced on a graph-edge e when mapping $\lambda_i f_i$ to G is at most $\lambda_i load_{T_i}(e)$.Therefore, the relative load induced on e when mapping all flows $\lambda_i f_i$ is at most $\sum_i \frac{load_{T_i(e)}}{c(e)} = \sum_i \lambda_i rload_{T_i}(e) \leq \beta$. □

Approximating the Bottlenecks Of a Graph By a Tree.

From the above two claims we have that given a multicommodity flow with congestion $C_{opt}(G)$ in G, and computing the flow in each tree T_i and scaling it by a factor λ_i we have a solution in G with congestion at most $\beta \cdot \max_i\{C_{opt}(T_i)\} \leq \beta \cdot C_{opt}(G)$.
At the last section we will prove the following theorem.

Theorem

For a graph G there exists a convex combination of decomposition trees T_i defined by multipliers λ_i with $\sum_i \lambda_i = 1$ such that the following holds. Suppose we are given for each tree T_i a multicommodity flow f_i that has congestion at most C. Then mapping the flows f_i to G while scaling flow f_i by λ_i results in a multicommodity flow $f := \sum_i \lambda_i m_{T_i}(f_i)$ in G that has congestion at most $O(\log n)$.

Finding a Convex Combination of Trees.

Finding a convex combination of decomposition trees such that every edge has expected relative load at most β is relative of finding a point in the following Polyhedron $P(\beta)$:

$$\forall e \in E \sum_i \lambda_i rload_{T_i}(e) \leq \beta$$

$$\sum_i \lambda_i \geq 1$$

$$\forall i \; \lambda_i \geq 0$$

Finding a Convex Combination of Trees.

We can write the above polyhedron in matrix form. Indeed let $\mathcal{T}$ denote the set of all decomposition trees, and let M denote an $|E| \times |\mathcal{T}|$ matrix with $M_{eT} = rload_T(e)$. Then we can write the edge-constraints from above as $M \cdot \vec{\lambda} \leq \beta \cdot \vec{1}$.

We define $lmax(\vec{x}) := \ln(\sum_e e^{x_e}) \leq \max_e x_e$. We can now write the above polyhedron as:

$$lmax(M\vec{\lambda}) \leq \beta$$

$$\sum_i \lambda_i \geq 1$$

$$\forall i \; \lambda_i \geq 0.$$

Finding a Convex Combination of Trees.

We now give the above definitions:
We define a function

$$partial'_e(\vec{x}) := \frac{\partial lmax(\vec{x})}{\partial x_e} = \frac{e^{x_e}}{\sum_e e^{x_e}}.$$

$$partial_i(\vec{\lambda}) := \frac{\partial lmax(M\vec{\lambda})}{\partial \lambda_i} = \sum_e rload_{T_i}(e) \cdot partial'_e(M\vec{\lambda}).$$

Lemma

For all $\vec{x}, \vec{\epsilon} \leq 0$ *with* $0 \leq \epsilon_e \leq 1$, $lmax(\vec{x} + \vec{\epsilon}) \leq lmax(\vec{x}) + 2\sum_e \epsilon_e partial'_e(\vec{x})$.

Finding a Convex Combination of Trees.

Lemma

For some decomposition tree T_i we use $l_i := \max_e\{rload_{T_i}(e)\}$ to denote the maximum load induced on a link in G. Let $\vec{\delta} = \delta_i \vec{e}_i$ with $\delta_i \leq \frac{1}{l_i}$. Then,

$$lmax(M(\vec{\lambda} + \vec{\delta})) \leq lmax(M\vec{\lambda}) + 2\sum_e (M\vec{\delta})_e partial'_e(M\vec{\lambda}) = lmax(M\vec{\lambda}) + 2\delta_i partial_i(\vec{\lambda}).$$

This means that if we have a tree T_i that has $partial_i(\vec{\lambda}) \leq \beta$ then increasing the variable λ_i by $\delta_i \leq \frac{1}{l_i}$ causes $\sum_i \lambda_i$ to increase by δ_i, while $lmax(M\vec{\lambda})$ only increases $2\delta_i\beta$. Repeating this until $\sum_i^{\lambda_i}$ is larger than 1 gives a set of variables λ_i that fulfill the constraints in Polyhedron $P(\beta)$

Finding a Convex Combination of Trees.

Lemma

For a vector $\vec{\lambda}$ of multipliers with a polynomial number of non-zero entries we can efficiently compute a tree T_i such that $partial_i(\vec{\lambda}) = O(\log n)$.

Απόδειξη.

For a tree T_i we have $partial_i(\vec{\lambda}) = \sum_e rload_{T_i}(e) \frac{e^{(M\vec{\lambda})_e}}{c(e) \sum_e e^{(M\vec{\lambda})_e}}$ If we define $l(e) := \frac{e^{(M\vec{\lambda})_e}}{c(e) \sum_e e^{(M\vec{\lambda})_e}}$ to be the length of an edge, then finding the tree T_i such that the above is minimized is the MCCT problem where the requirements are $r(u,v) = c(u,v)$. Applying theorem 1 gives a solution with total cost at most $O(\log n) \cdot \sum_e r(e) \cdot l(e) \leq O(\log n) \cdot \sum_e c(e) \cdot \frac{e^{(M\vec{\lambda})_e}}{c(e) \sum_e e^{(M\vec{\lambda})_e}} = O(\log n)$ as desired. □

Finding a Convex Combination of Trees.

We now give the algorithm for finding the convex combination.
find convex combination()
for all $i : \lambda_i := 0$
while $\sum_i \lambda_i < 1$ do
find a tree T_i with $partial_i(\vec{\lambda}) \leq O(\log n)$
$l_i := \max_e\{rload_{T_i}(e)\}$
$\delta_i := \min\{l_i, 1 - \sum_i \lambda_i\}$
$\lambda_i := \lambda_i + \delta_i$
return $\vec{\lambda}$

Finding a Convex Combination of Trees.

Lemma

The number of iterations is $O(|E| \log n)$

Απόδειξη.

Define a potential function $\sum_e \sum_i \lambda_i rload_{T_i}(e)$ that describes the total relative load that has been placed on the edges so far. The potential function is bounded by $O(\log n) \cdot |E|$ as the relative load induced on an edge does not exceed $O(\log n)$ in the end. For an iteration of the algorithm let $e^{'}$ denote the edge that has the larger relative load for the chosen tree T_i. We increase λ_i by $\frac{1}{l_i}$. Hence $\sum_i \lambda_i rload_{Ti}(e')$ increased by 1 which in turn means that the potential function increases by 1. This gives a bound of $O(|E| \log n)$ on the number of iterations. □