CS: 15-853 Algorithm in the real world **October 21, 2002**

Logistics 2: Approximation Algorithm for DD Problem

Lecture: R. Ravi Scribe: Zhong Xiu

Outline: BB and DD are two network design problems that often arise in practice. The two problems are in fact equivalent up to a small loss in the value of the solution. This lecture focuses on the approximation algorithms of deep-discount problem.

1 Motivation

Consider a network consisting of a single server and several clients. Each client wishes to route a certain amount of flow to the server. The cost per unit flow along an edge is proportional to the edge length.

DD (deep-discount): We call the problem of finding a minimum cost network supporting the required flow the *deep-discount* problem.

BB (buy-at-bulk): Alternatively, at each edge we might be able to pay for and install a certain capacity, and then route flow for free. The problem of finding a minimum cost network in this scenario is called the *buy-at-bulk* network design problem.

2 Formulation

Given: Graph $G=(V,E)$ with edge lengths l_e; a single sink $t \in V$; Sources $\{v_i\}_1^S = S \subset V$ with demands $\{dem_i\}_1^S$.

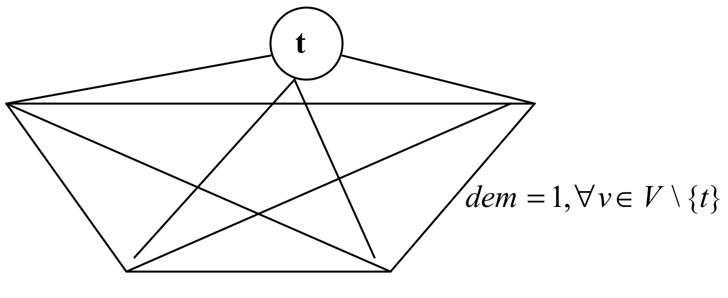

Figure 1: An Example of the Single Sink Deep Discount Problem's Graph

Given: k cable types, with characteristics (p_i, r_i). (**P**rice (of installation), **R**ate (of routing unit flow)) per unit length.

Goal: Install cables $i(e)$ on edges e to route flow at minimum cost:

$$\sum_{e\in E} l_e(p_{i(e)} + f_e r_{i(e)})$$

where f_e is flow on edge e.
This is the single sink deep discount problem.

3 Assumptions for DD (w.l.o.g.)

1. Assume triangle inequality holds for edge lengths in graph. (Edge length on metric)

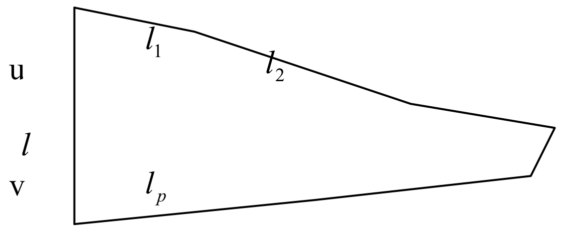

Figure 2: Triangle Inequality Assumption
$l > l_1 + l_2 + ... + l_p$

For approximation, assume:
2. Prune cable types to retain only those with sufficient decrease in routing cost. That is,
$r_{i+1} \le (1-\varepsilon) r_i, \forall i$ remaining.
e.g.
1: p_1, $r_1 = 1$;
2: p_2, $r_2 = 0.7$;
3: p_3, $r_3 = 0.4$;
Here $\varepsilon = 0.5$, so the second one is removed.

3. Prune further to retain only types with sufficient increase in price. That is,
$p_{j+1} \ge \frac{1}{1-\varepsilon} p_j, \forall j$ remaining.

EX1: Show 2 pruning steps still "leave" an optimal solution that is $O(1)$ times original optimal solution.

Proof:
Consider the goal: $\sum_{e\in E} l_e(p_{i(e)} + f_e r_{i(e)})$

Assume in the original graph, for a particular $p_{i(e)} + f_e r_{i(e)}$, p_i, r_i make an optimal solution and has been pruned in our pruning steps. Let the nearest remaining one is p_{i-1}, r_{i-1} ($p_{i-1} > p_i, r_{i-1} < r_i$).

Let $K = \frac{p_{i-1}}{p_i}$, we have $r_i > (1-\varepsilon) r_{i-1}$.

So $p_{i-1} + f r_{i-1} = K p_i + f r_{i-1} < K p_i + f \frac{1}{1-\varepsilon} r_i \le \max(K, \frac{1}{1-\varepsilon})(p_i + f r_i)$.

If we choose $K = \max(\frac{p_{i-1}}{p_i}), \forall i$, we can bound our solution that is $\max(K, \frac{1}{1-\varepsilon})$ times original optimal solution, that's $O(1)$.

4 Two lower bounds

Consider two natural lower bounds for the single-sink single-cable-type version.

All sources must reach the sink $\Rightarrow$ network must connect sources to sink $\Rightarrow$ minimum price of a connecting network is a lower bound.

Every source must route its flow to sink $\Rightarrow$ routing cost along shortest path rooted at sink is a lower bound.

So we get 2 lower bounds:

1. *Connectivity:* $\exists$ path from all sources to sink.

$\Rightarrow \cos t \ge$ (Minimal length of a tree connecting $S \cup \{t\}$) $p_{(\min)}$.

2. *Routing:* all flows must go on a path, at least as long as a shortest path to t $\Rightarrow \cos t \ge (\sum_{v \in S} dist^{1}(v,t) dem(v)) r_{(\min)}$.

5 Spanners for a Single Sink: LAST

5.1 LAST algorithm

Distance-preserving lightweight sub-graphs are called Spanners in the literature.

E.g. *LAST* or *Light Approximate Shortest Path tree*: For an n-node undirected edge-weighted graph and any $\alpha > 1$, a near-linear-time algorithm gives a tree such that:

a). Its weight at most α times that of an MST.

b). The distance from any node to the sink is at most $\frac{2}{\alpha - 1} + 1$ times that in the original graph.

That's to say:

$\forall G, l$ (Not necessary metric): $E \to \Re^+$, $\exists T$

a). $dist_T(v,t) \leq \alpha \times dist_G(v,t) \quad \forall v \in V \setminus \{t\}$.

b). $l(T) \leq \beta l(MST(G,l))$ for $\beta \geq \dfrac{2}{\alpha - 1} + 1$

This is the (α, β) algorithm.

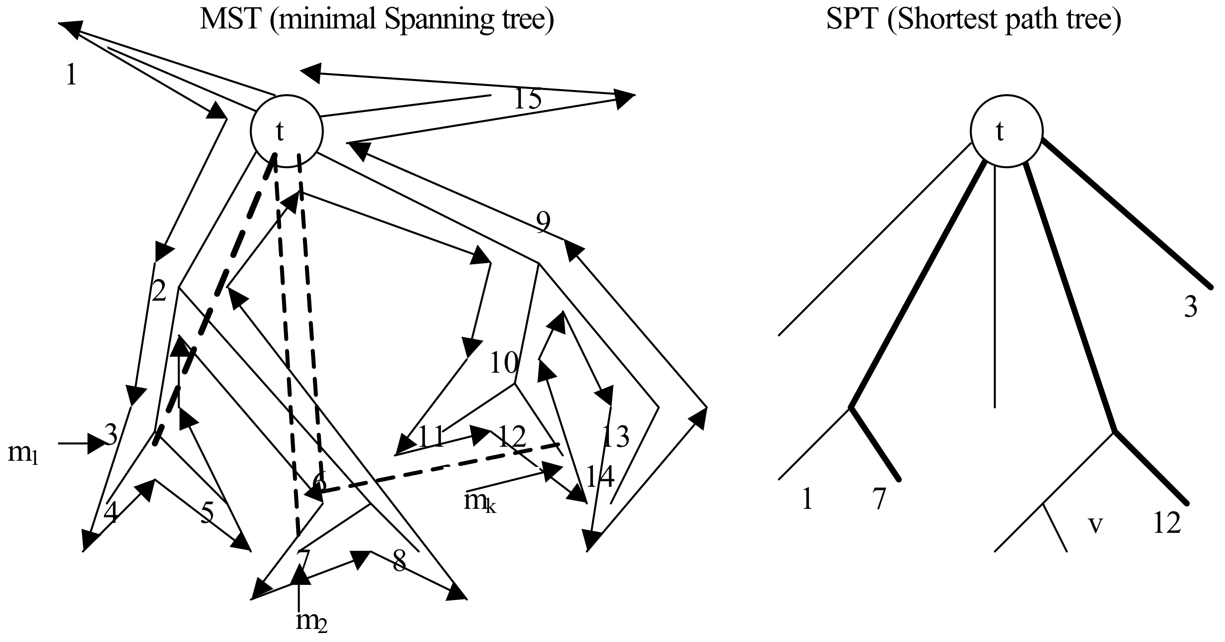

Figure 3: LAST Algorithm
The left tree is a minimal spanning tree
The right one is a shortest path tree

5.2 Analysis of Algorithm

Key idea: Find a LAST and route all flows along shortest paths in the LAST.
Analysis: Charge price of the cables in the LAST to the connection lower bond and the routing cost to the routing lower bound. Final performance guarantee is the sum of the two guarantees on the LAST.

The LAST algorithm goes as follows:

```
STEP 1: Find a minimal spanning tree (MST) of G;
            Find a shortest-paths tree (SPT) of G, with start node t;
STEP 2: Find a preorder numbering of MST using t as the start node;
STEP 3: $G_0 \leftarrow MST$
            for each node v in the preorder sequence do
                  Find a shortest t-v path P in SPT;
                  If $dist_{G_{i-1}}(v_i,t) > \alpha \times dist_G(v_i,t)$
                  Then add all the edges in a shortest t-v path in G to it;
                            That's to say: add shortcut $dist_G(v_i,t)$.
                            Update $G_i \leftarrow G_{i-1} + shortcut(v_i,t)$.
                  end;      (if)
          end; (for)
STEP 4: find a shortest-paths tree: $SPT(G_{final},t)$
```

5.3 Performance Ratio

Now analyze the upper bound of this algorithm.

$$l(T) \le l(G_{final}) = l(MST) + \sum_{m_i} dist_G(m_i,t)$$

$$\forall j, dist_{G_{j-1}}(m_j,t) > \alpha \times dist_G(m_j,t)$$

$$or \qquad dist_{MST}(m_j,m_{j-1}) + dist_G(m_{j-1},t) \ge dist_{G_{j-1}}(m_j,t) > \alpha \times dist_G(m_{j-1},t)$$

$$\sum_j 2l(MST) \ge \sum_j (\alpha - 1) dist_G(m_j,t).$$

This implies $l(T) \le \beta l(MST(G,l))$.

For the two costs, we have:

1. $Cost_{connectivity} \ge l(MST)p$
2. $Cost_{route} \ge \sum_i dist_G(v_i,t)r$ (assume dem=1)

And for the two parts of the final total cost, we have:

1. $p(T) \le \beta l(MST)p$
2. $route\text{-}cost(T) = \sum_i dist_T(v_i,t)r \le \alpha \sum_i dist_G(v_i,t)r$

So finally we'll have:

$$Total - cost(T) = p(T) + route - cost(T) \le \alpha Cost^*_{connectivity} + \beta Cost^*_{route} \le \max(\alpha,\beta) Cost^*_{total}$$

This gives the upper bound for the performance ratio.

EX2: If ρ_{ST} is the performance ratio for Steiner Tree problem, compute best performance ratio for DD (SC) as a function of ρ_{ST} .

6 General solutions methods for DD

Two methods are discussed:

1. LP rounding

2. "Direct" randomized method.

6.1 LP rounding algorithm

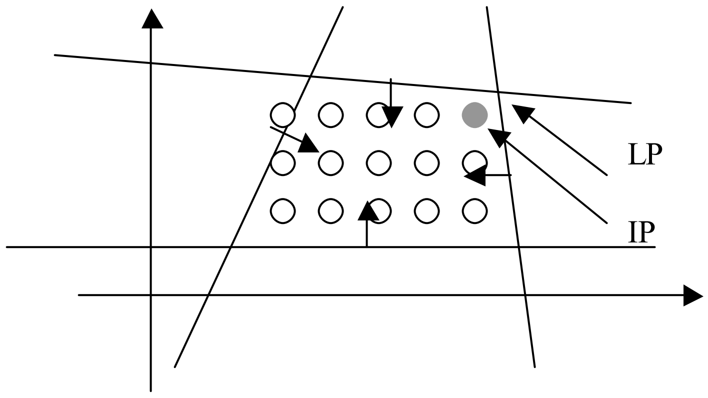

Figure 4: IP From LP Rounding
"Round" a solution to the Linear Programming relaxation of
an Integer Programming formulation of the problem

1. Go top down, starting with the sink t (and "thickest" cable type), and finishing with a tree that connects all sources.

2. K iterations: At each iteration, choose a subset of vertices and connect them using the next lower order cable type.

3. Last iteration: Connect any remaining sources with null (lowest order) cable type.

4. Routing: For each source, route along the unique source-sink path in this tree.

5. Using LP and LAST.

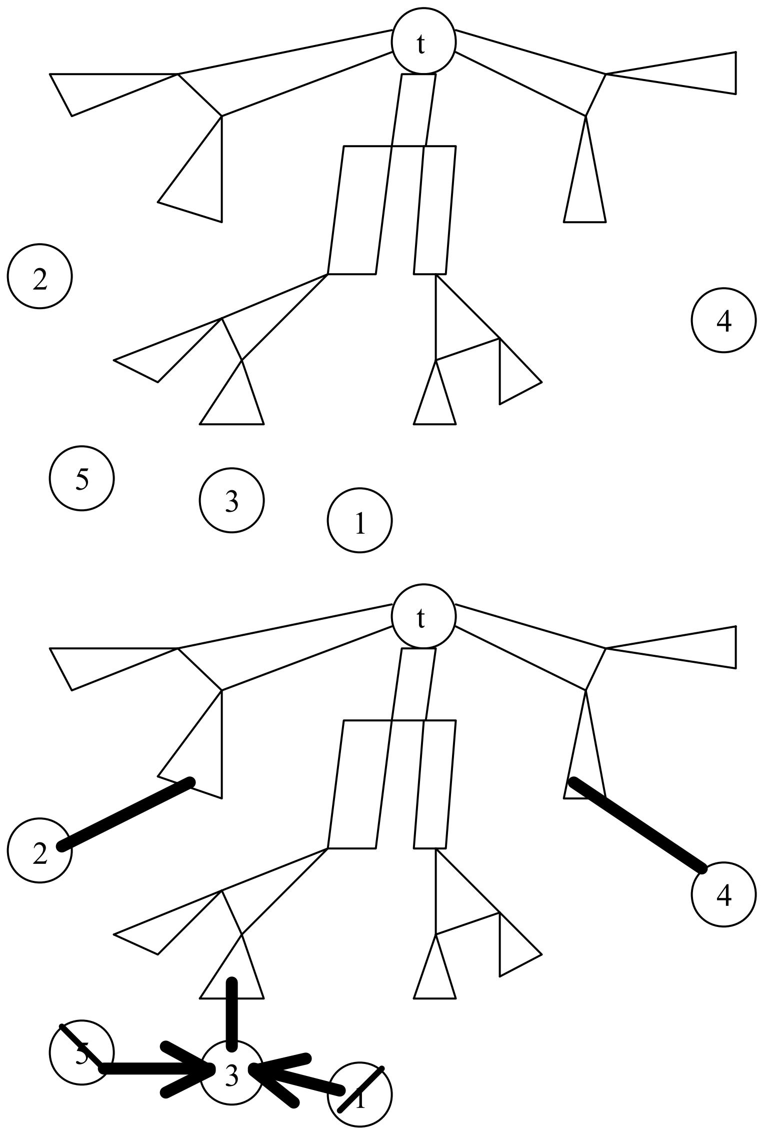

Figure 5: LP Rounding Problem

6.2 Direct Randomized Method

6.2.1 Key Ideas

1. Build solutions in increasing order of cable type (bottom up).

2. Use optimal solution to bound cost of partial solutions built at each iteration.

3. Two phases per iteration: one for connectivity and the other for routing, roughly.

4. Randomized redistribution steps after connectivity and routing phases ensure that expected demand at each source stays unchanged through the iterations.

6.2.2 Motivations

Run iterations for cable type i when enough demand has been aggregated at hubs so that cable of type i is economically preferable:

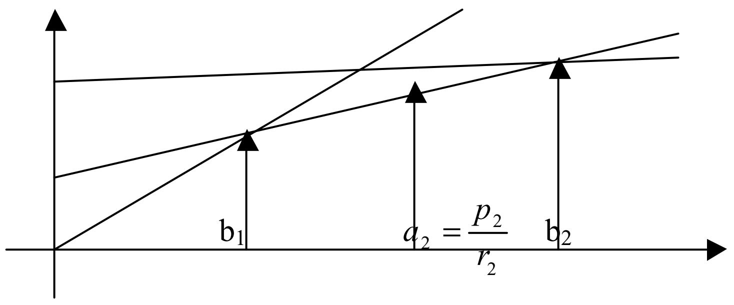

Figure 6: Breakeven Point

Breakeven point b_i, for type i is the smallest flow for which type i cable is better than lower type, i.e.,
Min f s.t.

$$p_{i-1} + fr_{i-1} \geq p_i + fr_i$$

Simplifying

$$b_i = \frac{p_i - p_{i-1}}{r_i - r_{i-1}}$$

However, until a certain Aggregation point when the routing cost is equal to the price, flow must be routed like a Steiner tree since the price dominates.

$$a_i = \frac{p_i}{r_i}$$

Once has been aggregated, flow must be routed along Shortest Paths until enough demand of up to the next breakeven amount collects.

6.2.3 Overview

At the start of iteration i, b_i or more demand has inductively been collected at a subset of demand points. In this iteration,

Build a Steiner tree with current cable type i.

Truncate tree to bunch demands to and aggregate to a_i one source randomly.

Route aggregated demands to consolidation points by solving a lower-bounded facility location problem with lower bound b_{i+1}.

Redistribute the consolidated demand randomly back to one of the originating sources.

Reference:

1. Naveen Garg, Rohit Khandekar, Goran Konjevod, R. Ravi, F.S.Salman, Amitabh Sinha, "On the Integralty Gap of a Natural Formulation of the Single-Sink Buy-at-bulk Network Design Problem", *Proceedings of the Conference on Integer Programming and Combinatorial Optimization* (IPCO 2001)

2. Slides From a Talk: "Approximation Algorithms for Buy-at-bulk Network Design", R. Ravi