Network Design and Game Theory
Spring 2008
Lecture 6

Guest Lecturer: Aaron Archer
Instructor: Mohammad T. Hajiaghayi
Scribe: Fengming Wang

March 3, 2008

1 Overview

We study the Primal-dual, Lagrangian relaxation paradigm of approximation algorithms for the NP-hard problems: Steiner tree problem (both the standard and prize-collecting version) and k-Minimum spanning tree problem. Its corresponding reference is [1].

2 General Steiner Tree

2.1 Definition

Given the graph $G = (V, E)$, the cost function $C : E \to \mathbb{R}^+$ and a subset of terminal nodes $\mathcal{T} \subseteq V$, Find $T \subseteq E$ that spans $\mathcal{T}$ of the total minimum cost. For example, let $\mathcal{T} = \{a, b, c\}$ in Figure 1. The extra node c is called a *steiner node*. It is easy to see that the corresponding steiner tree is the star centered at c of total weight 3 while any other spanning tree without using c costs 4 units.

Without loss of generality, we assume that G is always a complete graph, moreover, C obeys triangle inequalities, since the cost function could be transformed into a metric by taking the shortest path as the distance between any two nodes.

2.2 2-Approximation via MST

Compute the minimum spanning tree MST on the subgraph ~~ph~~ induced by $\mathcal{T}$, we claim that this 2-approximates the optimal steiner tree OPT. The proof consists of two steps. Since OPT is a tree, one could do the depth-first search on it from an arbitrary node, which constructs a Eulerian tour

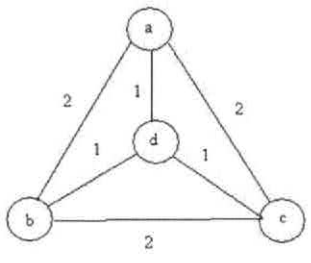


Figure 1: Steiner Tree

on the nodes spanned by OPT. Note the cost is doubled for the tour compared to OPT. Then revise the tree by skipping all of the steiner nodes and joining consecutive terminal nodes according to the tour, which returns a spanning tree on $\mathcal{T}$. By triangle inequalities on C, the cost of this spanning tree is no larger than that of the tour, therefore, MST is a valid 2-approximation.

2.3 2-Approximation via LP

First we fix some notations.

- For any $S \subseteq V$, $\delta(S) = \{e_{u,v} \in E : u \in S, v \notin S\}$.
- $\mathcal{C} = \{S \subseteq V : S \cap \mathcal{T} \neq \emptyset, \neq \mathcal{T}\}$.
- For any $F \subseteq E$, $C(F) = \Sigma_{e \in F} C_e$.

The following observation is easy to verify and crucial for our algorithm.

Claim 1 $T \subseteq E$ *spans $\mathcal{T}$ if and only if* $\forall S \in \mathcal{C}, T \cap \delta(S) \neq \emptyset$.
connect $\mathcal{T}$

Proof: Apply max-flow/min-cut theorem. ■

We interpret this idea in terms of integer programming and define the following indicator variables for each edge: $\forall e \in E$, $\chi_e = 1$ if e is in our solution tree, 0 otherwise.

$$\begin{array}{ll} \min & \Sigma_{e \in E} C_e \chi_e \\ \text{s.t.} & \Sigma_{e \in \delta(S)} \chi_e \geq 1 \quad \forall S \subseteq \mathcal{C} \end{array}$$

Relax the condition that $\chi_e \in \{0, 1\}$ to make them nonnegative real variables. Now we obtain a linear program and then take a close look at its dual.

$$\begin{array}{ll} \max & \Sigma_{S \in \mathcal{C}} y_s \\ \text{s.t.} & \Sigma_{S: e \in \delta(S)} y_S \leq C_e \quad \forall e \in E \\ & y_s \geq 0 \end{array}$$

Apparently both the primal and the dual programs are feasible. So the complementary slackness constraints are satisfied for the pair of optimal solutions.

1. $\forall e \in E, \chi_e > 0 \Rightarrow \Sigma_{S:e\in\delta(S)} y_S = C_e$.

2. $\forall S \in \mathcal{C}, y_S > 0 \Rightarrow \Sigma_{e\in\delta(S)} \chi_e = 1$.

These two implications have economic interpretations. The first postcondition says that for every edge, the primal buys exactly what the dual pays for it while the second one means that each cut pays exactly for one edge. Therefore, we get a sequence of equalities as follows:

$$\Sigma_{e\in E} C_e \chi_e = \Sigma_{e\in E} \chi_e \Sigma_{S:e\in\delta(S)} y_S = \Sigma_{S\in\mathcal{C}} y_S \underbrace{\Sigma_{e\in\delta S} \chi_e}_{1 \text{ whenever } y_S \neq 0} = \Sigma_{S\in\mathcal{C}} y_S$$

So any answer for the dual is always a lower bound for the primal. Next we would devise an algorithm which promises that the output never exceeds twice one specific dual solution.

Agrawal-Klein-Ravi (AKR) Algorithm For Steiner Tree

```
$\mathcal{F} = \phi$.
**repeat**
  **repeat**
    Grow at unit rate $y_S$ for every component induced by $\mathcal{F}$ that contains a terminal.
  **until** Some edge $e$ goes tight (is paid for by the cut it crosses)[Break ties arbitrarily].
  $\mathcal{F} \leftarrow \mathcal{F} + e$ [This will never create a cycle because edge $(u, v)$ stops being paid off once $u$&$v$ are in the same component].
  Add edges without creating cycles and ties are broken arbitrarily.
**until** $\mathcal{F}$ spans $\mathcal{T}$.
Pruning: iteratively delete edges to steiner nodes that are leaves.
```

Consider the graph in Figure 2, where the terminal nodes are a, b, e and f. The first several steps of the algorithm are performed as follows:

1. Grow $y_{\{a\}}, y_{\{b\}}, y_{\{e\}} y_{\{f\}}$ at rate 1. Merge ac, bd, eg at time 1.

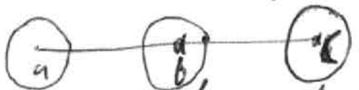
2. Grow $y_{\{a,c\}}, y_{\{b,d\}}, y_{\{e,g\}}, y_{\{f\}}$. Merge $aceg, bdf$ at time 2.

3. ...

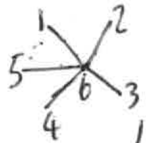
Remark: The y_S values produced by the algorithm always satisfies the constraints in the dual program. So they form a feasible solution.

Remark: The ratio between the optimal solution for the linear program and the output of the algorithm could be as large as 2. It is not hard to see that a simple cycle yields this gap.

Idea of the proof of Approximation factor 2 (Thm 1)

If we have path for node b 

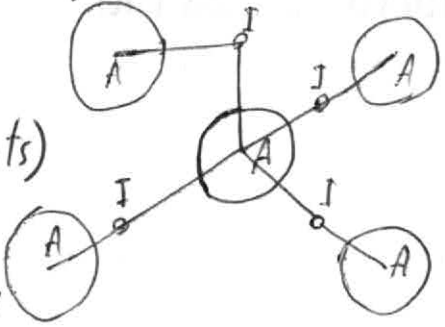
we increase Δ but indeed we buy 2Δ edges and this is factor 2 what about

if 

, then we pay in average 2Δ.

more precisely consider an snapshot of the algorithm whose final solution is F.

The set A is the set of growing balls (Active components)



consider I as the set of Inactive components (isolated steiner nodes which are in the final solution F)

Rate of paying for Primal is $\sum_{V\in A} deg(V)$ while the rate of increasing dual is |A|

Since All leaves are active nodes (steiner nodes will be useless as leaves due to pruning step that we delete unnecessary edges) the inactive nodes have degree at least 2 thus $\sum_{V\in I} deg(V) \geq 2|I|$ (*)

we have $\sum_{V\in A\cup I} deg(v) = 2(|A|+|I|-1)$ since the final solution is a tree.

Due to (*) thus $\sum_{V\in A} deg(V) \leq 2(|A|-1) \leq 2|A|$. Thus $\frac{d(\text{primal increas})}{dt} \leq 2\frac{d(\text{dual incre}}{dt}$

Thus by induction $cost(F) = \text{cost of primal} \leq 2 \text{ cost of dual} \leq 2opt$ ✓

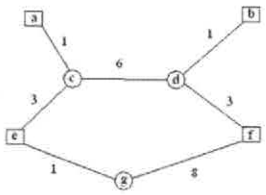


Figure 2: Example

Theorem 1 *$\mathcal{F}$ output by the algorithm satisfies $C(F) \leq 2(1 - \frac{1}{n})\Sigma y_S$, where the y_S values untouched by the algorithm are implicitly set to 0.*

The proof does the backward analysis and utilizes the fact that in any tree, the total degree never exceeds twice the number of nodes. For more details, we refer the reader to chapter 22 of the book "Approximation Algorithms".

3 Prize-collecting Steiner Tree (PCST)

This is similar to the problem previously studied. In addition, there two more parameters, $\Pi : V \to R^+$, the prize function which associates a nonnegative value to each node once reached, and $r \in V$, the root from which one tries to build the steiner tree. The goal is changed as well: we want to construct a tree T rooted at r such that the gain (the difference between the sum of prizes from the connected nodes and the total cost to realize the tree) is maximum. Formally the following quantity needs to be maximized:

$$\Pi(T) - C(T) \tag{1}$$

which is equivalent to maximize

$$\Pi(T) - C(T) - \Pi(V) \tag{2}$$

or

$$-C(T) - \Pi(V \setminus T) \tag{3}$$

Remark 1 *This is the same as minimizing $C(T) + \Pi(V \setminus T)$. This seems like an unnatural objective function. The original motivation for studying it is that it is NP-hard to determine whether the optimum value for 1 is positive, and hence* *no bounded approx ratio is possible unless $P = NP$. In contrast, we can get a 2-approx for 3, as shown below. This algorithm has a "Lagrangian multiplier-preserving" property that turns out to have both theoretical and practical consequences that we will exploit.*

For every $U \subseteq V \setminus \{r\}$, z_U equals to 1 if U is exactly the set we don't span, 0 otherwise. Therefore, an easy observation leads to the following integer program.

$$\begin{array}{ll} \min & \Sigma_{e\in E} C_e \chi_e + \Sigma_{U\subseteq V\setminus\{r\}} \Pi(U) z_U \\ \text{s.t.} & \Sigma_{e\in\delta(S)} \chi_e + \Sigma_{U\supseteq S} z_U \geq 1 \qquad \forall S \subseteq V\setminus\{r\} \\ & \chi_e, z_U \geq 0 \end{array}$$

The corresponding dual is

$$\begin{array}{ll} \max & \Sigma y_S \\ \text{s.t.} & \Sigma_{S: e\in\delta(S)} y_S \leq C_e \quad \forall e \in E \\ & \Sigma_{S\subseteq U} y_S \leq \Pi(U) \quad \forall U \subseteq V \\ & y_S \geq 0 \end{array} \tag{4}$$

We have one new implication added into the complementary slackness.

$$z_U > 0 \Rightarrow \Sigma_{S\subseteq U} y_S = \Pi(U)$$

Goemans-Williamson Algorithm For PCST

```
$\mathcal{F} = \phi$.
$\mathcal{A} = \{\{v\} : \Pi_v > 0\}$
$\mathcal{I} = \{\{v\} : \Pi_v = 0\}$ [This includes $\{r\}$ since we assume $\Pi_r = 0$]
$\forall v \in V, P_{\{v\}} = \Pi_v$ [We call the P variables potentials.]
repeat
  repeat
    Grow $y_S$ and shrink $P_S$ for all $S \in \mathcal{A}$.
  until Some edge e goes tight or for some $S \in \mathcal{A}$, $P_S$ hits 0.[Ties are broken arbitrarily.]
  if Edge e goes tight then
    $\mathcal{F} \leftarrow \mathcal{F} + e$.
    Merge $S_1, S_2$ that e joins and let $P_{S_1\cup S_2} = P_{S_1} + P_{S_2}$
    Delete $S_1, S_2$ from $\mathcal{A}$ (or $\mathcal{I}$) and add $S_1 \cup S_2$ to $\mathcal{A}$.
    if $r \in S_1 \cup S_2$ then
      Move $S_1 \cup S_2$ from $\mathcal{A}$ to $\mathcal{I}$.
    end if
  else if For some $S \in \mathcal{A}$, $P_S$ hits 0 then
    move S from $\mathcal{A}$ to $\mathcal{I}$
  end if
until $\mathcal{A} = \phi$ or $\mathcal{F}$ spans the whole graph.
Pruning: throw away every component that was ever an inactive leaf.
```

scribe: Fengming Wang
Lecture 6 **Date:** 03/03/2008

Note 1 *GW should have to implement this using standard priority queue in* $O(n^2 \log n)$ *time.*

At the end of the algorithm, we have some tree T. Because the nodes we fail to span can be partitioned into components that ran out potential, and this was constructed to be equivalent to constraint 4 being tight for that component, we get

$$\Sigma_{S \subseteq V \setminus T} y_S = \Pi(V \setminus T)$$

Moreover, the analogous analysis in the last section gives

$$C(T) \leq 2\Sigma_{S: T \cap \delta(S) \neq \phi} y_S$$

Combine them together and we get

Theorem 2 $C(T) + 2\Pi(V \setminus T) \leq 2\Sigma y_S$

To have a 2-approximation for PCST, it would have been sufficient to show $C(T) + 2\Pi(V \setminus T) \leq 2\Sigma y_S$.

The two sides being equal means that the Lagrangian-preserving Property is satisfied. If we multiply the prize function by a parameter λ to find some tree $T(\lambda)$ via GW PCST algorithm, this is very useful which amounts to the following claim.

Claim 2 *Among all trees that capture at least* $\lambda\Pi(T(\lambda))$ *prizes, the cost of* $T(\lambda)$ *is no worse than twice that of the optimal.*

Proof: As the previous linear program

$$\begin{array}{ll} \min & \Sigma C_e x_e \\ \text{s.t.} & \Sigma_{e \in \delta(S)} x_e + \Sigma_{U \supseteq S} z_U \geq 1 \quad \forall S \subseteq V \setminus r \\ & \Sigma_{U \subseteq V \setminus r} \Pi(U) z_U \leq G \\ & x_e, z_U \geq 0 \end{array}$$

Its dual is

$$\begin{array}{lll} \max & \Sigma_{S \subseteq V \setminus r} y_S - \lambda G & \\ \text{s.t.} & \Sigma_{S: e \in \delta(S)} y_S \leq C_e & \forall e \in E \\ & \Sigma_{S \subseteq U} y_S \leq \lambda \Pi(U) & \forall U \subseteq V \setminus r \\ & \lambda, y_S \geq 0 & \end{array}$$

Think of G as $\Pi(V \setminus T(\lambda))$. Then similar to the previous analysis, $C(T(\lambda)) \leq 2(\Sigma_{y_S} - \lambda\Pi(V \setminus T(\lambda)))$. ■

The theorem above gives us a base for the algorithmic paradigm which "turns the λ knob" to search for a better approximation.

scribe: Fengming Wang
Lecture 6 **Date:** 03/03/2008

4 k-MST problem

Given $G = (V, E)$, the cost function $C : E \to R^+$, the root $r \in V$ and $k \in \mathbb{Z}^+$, find the tree T rooted at r spanning at least k nodes of the smallest total edge-cost.

Notation:

- $T(\lambda)$ = tree given by GW PCST run with ~~edge profit~~ Penalty λ. for each vertex
- OPT_k = cost of the optimal k-MST.
- LB_k = a valid lower bound on OPT_k

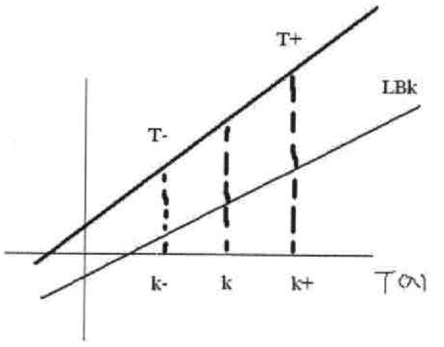


Figure 3: Search

The sketch of the idea is that find a value of λ (using binary search, Meggido parametric search, etc.) such that $|T(\lambda^-)| \leq k$ and $|T(\lambda^+)| > k$, where $|T(\lambda^-)|$ and $|T(\lambda^+)|$ are 2-approximate k'-MSTs for $k' = |T(\lambda^-)|$ and $k' = |T(\lambda^+)|$. If $k = |T(\lambda^-)|$ we are done and have a 2-approximate solution. Otherwise, we must combine the trees $T(\lambda^-)$ and $T(\lambda^+)$ to get our final solution T. If $c(T)$ are exactly the linear interpretation of $c(T(\lambda^-))$ and $c(T(\lambda^+))$, then we would have a 2-approximation. This is essentially how a 2-approximation is obtained, although it is possibly to get a 5 or a 3 much more simply.

5 Real world Application

At AT&T a team that includes Aaron has built a tool that uses the Goemans-Williamson PCST algorithm to help decide where to lay new cables. The tree produced by the GW algorithm is a first cut that is fed into some heuristics to augment the tree into a 2-connected network. We have found this approach to be practical and effective. The multiplier λ can be set to reflect the target payback period for this capital investment.

Real world AT&T application: Design fiber build connecting new customers to existing network. The graph is the street network.
Root: existing fiber (super node). Edge cost: cost of digging trench for laying fibers. Prize is the monthly income from a customer.
The prize-collecting steiner tree models the total <u>loss</u> of the company.

scribe: Fengming Wang
Lecture 6 **Date:** 03/03/2008

References

[1] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In *SODA*, pages 760–769, 2000.

Prize-collecting (PC) problem are classic optimization problems in which there are various demands that desire to be served by some lowest-cost structure. However, if some demands are too expensive to serve, then we can refuse and instead pay a penalty. Several applications both in theory and practice. Usually there is a constant factor (an additive factor 1 difference) for approx. factor of PC problems vs. non-PC (regular) problems (though there are some exceptions). Still we do not know can this additive +1 can be eliminated (we can improve it to +0.5 but not more so far.

PC-TSP 1.91 vs. 1.5 for TSP

PC-Steiner tree 1.96 vs. 1.39 for Steiner tree

PC-Steiner forest 2.54 vs 2 for steine forest

there are $\overset{\text{definition}}{\downarrow}$ a set of pairs, that we want to connect and a penalty for each one which is not connected.

An example of additive factor 1: consider PC-ST and write LP

$$LPOPT = \min \sum_{e\in E(G)} c_e x_e + \sum_{t\in T} z_t \pi_t$$

$$\text{s.t.} \quad \sum_{e\in\delta(S)} x_e + z_t \geq 1 \quad \forall_{t\in T}, S\subset V \text{ where } r\notin S \text{ and } t\in S$$

$$x_e \geq 0,\ z_t \geq 0.$$

Solve the LP and consider all $T' \subseteq T$ for which $z_{t \in T'} \geq \frac{1}{3}$. Pay the penalty for these t's. Then increase x_e by a scale factor $\frac{3}{2}$. Then we know for $t \in T - T'$, $\sum_{e\in\delta(S)} x_e \geq 1$ for any S where $r\notin S$ and $t \in S$. Then we know cost of connecting $T-T'$ is at most twice this LP (by primal-dual algorithm). Total cost is $\leq 2 \sum_{e\in E(G)} c_e (\frac{3}{2}) x_e + 3 \sum z_t \pi_t = 3\,OPT$.

We can improve 3 to 2.54 using a randomized algorith