CS 684 Algorithmic Game Theory	**Scribe: Prakash Linga**
Instructor: Eva Tardos	**9th April, 2004**

In this lecture we are going to talk about cost sharing in graphs. This is the last lecture on cost sharing. This class is about using primal-dual method for generating cross-monotone shares. Examples of this include:

- Spanning tree (easy version, handled today)
- Submodular functions (not handled in class, refer relevant papers)

Spanning tree as cost sharing

0.1 Problem definition

The problem setting is as follows. Clients want to connect to the server. Here, bandwidth is not an issue and hence clients can connect through each other to finally connect to the server. We are given:

- Server r (root)
- Set of players N (players, clients, users are used synonymously)
- Distances (or costs) on $r + N$ ($r + N \subseteq$ *nodes in* G or $r + N \subseteq$ *metric space*)
- User i has utility U_i

Problem: Select subset $S \subseteq N$ of users, build a min-cost network graph on $S + r$ and share costs between users in S

One interesting special case of a min-cost graph that we consider is the minimum spanning tree (MST). Let $c(S)$ denote the min-cost graph on $S + r$. Also, let $c_{MST}(S)$ be the MST cost on $S + r$.

0.2 Fact

$$c_{MST}(S) \geq c(S) \geq \frac{1}{2} c_{MST}(S)$$

$c_{MST}(S) \geq c(S)$ is obvious. $c(S) \geq \frac{1}{2} c_{MST}(S)$ is a well known result.

0.3 Facts from last class

1. Cost sharing is cross-monotone implies cost shares in core.
2. Cross-monotone cost sharing leads to a strategy proof way to select users and share cost.
3. Cost is submodular.

0.4 Properties of tree cost

We will show that tree cost is sub-additive: If A, B are two disjoint sets then, $c(A)+c(B) \geq c(A+B)$. Note that sub-additive property is necessary for cross-monotone cost sharing.

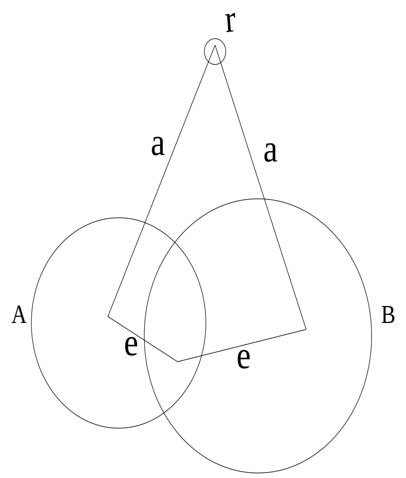

Figure 1: Lemma 2 illustration

Lemma 1

If cost function c admits a cross-monotonic cost-sharing than it must be subadditive. **Proof**. Let $\zeta_i(A)$ be the cost-share of user i in set A. Then we have that $c(A)+c(B) = \Sigma_{i \in A}\zeta_i(A) + \Sigma_{i \in B}\zeta_i(B)$ $\geq \Sigma_{i \in A \cup B}\zeta_i(A \cup B)$ (using cross-monotonicity of ζ_i).

Now, $\Sigma_{i \in A \cup B}\zeta_i(A \cup B) = c(A \cup B)$ (by definition) and hence $c(A) + c(B) \geq c(A + B)$. ■

Lemma 2

Tree cost is sub-additive i.e if A, B are two disjoint sets then, $c(A) + c(B) \geq c(A + B)$

Proof. As the two trees connecting A and B respectively, is a graph connecting $A \cup B$, possibly not the cheapest one. ■

Note

However that $c(S)$ and $c_{MST}(S)$ are not submodular.

Proof.

Considering the graph in Figure 1, $c(A) = a + \epsilon$, $c(B) = a + \epsilon$.

Also, $c(A \cup B) = a + 2\epsilon$. We can choose $c(A \cap B) > a$ so that $c(A \cup B) + c(A \cap B) > c(A) + c(B)$ and hence c is not always submodular. ■

We start by defining a simple cost-sharing. Given node i let $p(i)$ denote the parent of i in the spanning tree. Note that $p(r) = r$. Define cost share $\zeta_i(i)$ as the the cost of edge between i and $p(i)$.

Lemma 3

ζ_i is in core

Proof.

Let's assume ζ_i is not in core. Then let T_s be a cheaper tree on $S + r$.

Now replace edge $(i, p(i))$ for $i \in S$ by T_s. This leads to a cheaper tree. A contradiction.

■

0.4.1 Goal: Want cross-monotone shares for c_{MST}

Refer: Jain and Vazirani; Kent and Skorin-Kapov. The simpler analysis here due to Martin Pal.

0.5 Approximate computation

Note that it is NP-complete to compute $c(S)$ and that $c_{MST}(S)$ is an approximation to $c(S)$ within a factor of 2.

Our goal in this part is to compute cost shares for $c_{MST}(S)$ that are cross-monotone.

Definition ζ_i's are cost shares if $\Sigma_{i\in S}\zeta_i(S) \leq c(S)$

0.5.1 Computation of MST cost shares

We construct an MST using Kruskal's algorithm. The algorithm will maintain sets of nodes and each set will have a "reserve" r. The components will be connected sets. We will think of the reserve r as drawing a ball of radius r around the component. Throughout the algorithm all components will have the same reserve.

- Start with each node in its own component, and $\zeta_i = 0\ \forall i$, and $r = 0$. Think of r as the reserve "cash" of components, while ζ_i will be the cost-share of node i.
- While not all nodes are connected.
  - Increase component reserve r at the same rate for all components i. When $2r = c(i,j)$ (i.e when the balls of radius r around i and j touch) for two nodes i and j in separate components, build an edge, and merge the components containing i and j to a single component.

    `Charging scheme`: Use the reserve r of one component to "pay" for edge cost. Note that we only pay half the edge cost. The reserve of the other component is kept as reserve for the component. This maintains the invariant that all component reserves are the same. When one of the merged component is the root, than use the non-root component's reserve to pay for the edge.
  - `Cost-shares`: If connected component of node i has n_i nodes, and it does not contain the root, then increase the nodes cost-share at rate $\frac{1}{n_i}$. Do not increase cost-shares in the root component.

0.5.2 Properties

- Tree we build is a MST. This is because we used Kruskal's to obtain the tree.
- In any non-root component, the sum of all cost-shares equals the current reserve r plus the 1/2 of the total cost of all edges built inside the component.
- Components pay exactly for half the edge cost.
- The "reserve" of the root component is never used. (Which is good, since the nodes in the root component do not have cost-shares to pay for this!)

- Shares are cross-monotone. To see this note the following. More users implies rate at which ζ_i increases is slower. Components are connected components when everyone reaches out to some ζ distance. More users therefore implies growth stops earlier and hence the shares are cross-monotone ($\zeta_i(A) \geq \zeta_i(B) \forall A \subset B$).