"Group Strategyproof Mechanisms via Primal-Dual Algorithms"

Martin Pál and Éva Tardos (2003)

Key Points to Discuss

- The Cost-Sharing Problem
  - Metric Facility Location
  - Single Source Rent-or-Buy
- Definition of key properties
- Developing group strategyproof mechanisms for both problems using similar primal-dual algorithms.
- Approximation factors for both algorithms.

The Cost-Sharing Problem

Inputs: A set U of potential users. Each user has a private utility u_i of being serviced by a facility (or connected to a network).

Idea: We want to install or maintain a facility or network such that the users of the facility (network) share the cost of its installation (maintenance).

Outputs: A set $J \subseteq U$ of users that will be users of the facility (or part of the network). We also want the cost of installation or maintenance to be shared among the users so that everyone is "content" with the decision (known as *cost shares)*.

Meaning of “content”

(Properties of group strategyproofness)

A user should not have incentives to change his mind about contributing (or not contributing) to a facility or about being in (or not being in) the network later.

A user should not have incentives to lie about his utility.

A group of users should not be tempted to buy a separate facility or to build a separate network of cost smaller than the sum of their current contributions.

Key Properties of Cost Shares

Cross-monotonicity: The cost share of each user never goes up as the set of participating users increases.

Moulin and Shenker (2001) show that cross-monotonic cost sharing leads to group strategyproof mechanisms for determining the set of users (J).

Competitiveness: The sum of the cost shares cannot be more than the true cost.

Cost Recovery: The sum of the cost shares must pay for the true cost.

α-Approximate Cost Recovery: Users are only required to recover (pay for) $1/\alpha$ fraction of the true cost.

Developing a Group-Strategyproof Mechanism for the Cost Sharing Problem

We need to:

1. Determine the set J of participating users.

2. Determine cost shares with the properties:

   - Cross-Monotonicity,
   - Competitiveness, and
   - α-Approximate Cost Recovery.

Focus on two problems: Metric Facility Location (easy) and Single-Source Rent or Buy (more difficult).

Metric Facility Location

(Review from class)

Inputs: A set of potential facilities F and a set $J \subseteq U$ of users.

Idea: Open a subset of the facilities, paying amount f_p for each open facility p, and build a link from each user $j \in J$ to some open facility, given the cost of connecting user j to facility p defined as c_{jp}.

Outputs: Which facilities to open. Which users are connected to which facilities.

Facility Location Primal and Dual Formulations

Primal

$$\min \quad \sum_p f_p y_p + \sum_j \sum_p c_{jp} x_{jp}$$

$$\text{s.t.} \quad \sum_p x_{jp} \geq 1 \quad \forall j$$

$$y_p - x_{jp} \geq 0 \quad \forall j, \forall p$$

$$x_{jp} \in \{0, 1\}, \quad y_j \in \{0, 1\}$$

Dual

$$\max \quad \sum_j \alpha_j$$

$$\text{s.t.} \quad \alpha_j - \beta_{jp} \leq c_{jp} \quad \forall j, \forall p$$

$$\sum_j \beta_{jp} \leq f_p \quad \forall p$$

$$\alpha_j \geq 0, \quad \beta_{jp} \geq 0$$

Primal-Dual Algorithm for FL

Key Concepts

Grow a ball ("ghost") around each user until either (1) the ghost touches a full facility, or (2) a facility that the ghost is already touching becomes full.

When facility p becomes full we open it iff there is no already open facility q such that $c(p,q) \leq 2t(p)$ where $t(p)$ is the time when p becomes full. **This is slightly different than the method we discussed in class, but achieves the same approximation**.

The cost of the solution constructed is at most 3 times the sum of the cost shares. Thus the sum of the costs that a group of users pays to open a facility (cost shares) is at least 1/3 of the cost of opening the facilities plus connection costs.

Primal-Dual Algorithm for FL

Key Concepts, con't

Cross-Monotonicity: The cost share of each user never goes up as the set of participating users increases. By adding more users, each facility can only get filled more quickly, and hence each individual user can only become satisfied earlier.

Competitiveness: The method cannot charge users more than the true cost. The algorithm provides a lower bound to optimum.

α-Approximate Cost Recovery: As stated previously, this algorithm recovers 1/3 of the cost of the solution constructed.

MAIN RESULT: This primal-dual algorithm is a 3-approximate cross-monotonic, cost-sharing (i.e. group strategyproof) method for metric facility location.

Single-Source Rent or Buy

Inputs: A set of $J \subseteq U$ of users and a source s, residing in a graph $G = (V, E)$. A parameter M.

Idea: Build a tree such that there exists a path between each user and the source using edges in E. Each edge can either be bought at a cost Mc_e or rented at a cost of c_e. Bought edges can support any number of paths. The rental cost c_e must be paid for every path that uses edge e.

Outputs: A list of bought edges and rented edges.

Structure of the Optimal Solution

It is not hard to see that:

- The *bought edges* in the optimum rent or buy network must form a Steiner tree with s at its root.

- The *rental edges* form a shortest path connection from each user j to the closest point in the tree.

Primal and Dual Formulations

$$\min \sum_j \sum_i c_{ij} x_{ij} + M \sum_{e \in E} c_e z_e$$

s.t.

$$\sum_{i \in V} x_{ij} = 1 \forall j$$

$$\sum_{e out of S} Z_e \geq \sum_{i \in S} x_{is} - \sum_{i \in S} x_{it} for all S \subseteq V$$

$$\sum_{e out of S} Z_e \geq \sum_{i \in S} x_{is} - \sum_{i \in S} x_{it} for all S \subseteq V$$

One Algorithm Idea

- Grow a ghost around every user until the user's ghost touches a full center or a center the ghost is touching becomes full (as in FL).

- A location is full if it is touched by M or more users' ghosts. Since there are no opening costs, we open a center at any full location p.

- Use the primal-dual Steiner tree algorithm to connect the opened centers.

- Cost shares: Each user pays for its rental edges to connect to a center. Share the costs of buying edges to a center among the users of that center.

Primal-Dual Steiner Tree Algorithm

- Start with each center in a separate component.

- Grow a ghost around each center.

- When two ghosts touch:
  - Check if the two centers are in the same component.
  - If NOT, buy a shortest path between the ghost centers, merging the two components into one.
  - The center p of the first ghost to reach the source s is allowed to buy a shortest path between p and s.

- The algorithm incurs a cost M per time unit for growing each component that doesn't contain the root vertex s.

Problems

1. If we open all full centers many more centers are opened than necessary.

2. The cost of building a Steiner tree on all centers may be very large.

3. Each user may be connected to multiple centers.

4. Making decisions on how to allocate users to centers is necessarily influenced by the addition of new users. It is hard to guarantee cross-monotonicity.

A Better Idea

Simultaneously grow ghosts around users (to determine rental edges) and build a Steiner tree (to determine which edges to buy).

Simplifying Assumption: It is convenient to think of an edge e as a line segment of length c_e containing a continuum of points. The term *location* refers to both original vertices and intermediate points.

It has been shown that a solution that is allowed to rent and buy edge segments arbitrarily can be transformed into a solution that does not use intermediate points of no greater cost.

Definitions

$B(j,t)$: the ghost around j with radius t.

C: the (time-varying) set of all locations that have been reached by M or more user ghosts. At any time, C can be represented as a collection of vertices, edges, and segments of edges.

Let C $\subseteq C$.

Connected component of C: any inclusion-maximal C $\subseteq C$ such that any two location $p_1, p_2 \in$ C are joined by a path lying completely within C.

We say that a user j at time t is *connected* to a component C of C if $B(j,t) \cup$ C $\neq \emptyset$.

A user j is *satisfied* if it is connected to a component containing the source vertex s.

$t(j)$: the time when user j first becomes connected to a component.

$t'(j)$: the time when user j becomes satisfied.

Algorithm

The algorithm maintains a list of components C. Initialize: $C = \emptyset$.

Begin growing a ghost around each user. As time progresses, a user's ghost may reach another user's location.

Let S be this set of users that have reached a location p. When $|S| \geq M$, we add the set $\cap_{j \in S} B(j, t)$ of locations as a new component of C. At this time, location p is consider to be full.

We declare a full p open if at time $t(p)$ there is no open center within a radius $2t(p)$ of p.

As time progresses, two components may touch. When this occurs, we join the components by constructing a shortest path between their centers.

We continue until all users are satisfied and C contains a single component.

Cost Shares

We should make every user pay for the rental costs associated with him. If a user is connected to multiple components, we only let him contribute to the component where his share is the smallest.

Let $J(\mathrm{C})$ denote the set of users connected to C.

For a connected user j, let $a_j(t)$ be the maximum size $|J(\mathrm{C})|$ over all components C that j is connected to at time t.

Define f_j as follows: $f_j(t) = 1$ for $0 \leq t < t(j), f_j(t) = M/a_j(t)$ for $t(j) \leq t < t'(j)$, and $f_j(t) = 0$ for $t > t'(j)$.

Define α and α' for each user as follows:

$$\alpha_j = \int_0^{t'(j)} f_j(t)dt \text{ and } \alpha'_j = \int_0^{t(j)} f_j(t)dt = t(j).$$

Maintaining Key Properties

Theorem (Cross-Monotonicity). *The cost shares α and α' are cross-monotonic functions of the set J.*

Proof sketch. By adding more users we can only make the set C larger, and hence make each user connect earlier. Moreover, each component of C can only increase by adding more users. Hence, as the number of users connected to that component increases the cost shares can only grow slower.

Theorem (Competitiveness). *Every feasible solution to the single source rent or buy instance has a cost of at least* max $(1/2 \sum_{j \in J} \alpha_j, \sum_{j \in J} \alpha'_j)$.

α-Approximate Cost Recovery

Lemma. *Let p be an open center, and let $j \in S_p$. Then $3\alpha'_j \geq t(p)$. For a j that does not belong to any S_p, there is an open ghost center p such that $c(j,p) \leq 3\alpha'_j$.*

This lemma shows that the cost shares α' can pay for 1/3 of the rental cost of the network (similar to the facility location case).

Lemma. *The cost of the tree we buy is at most $6\sum_j \alpha_j$.*

Bounding the tree cost (next slide).

Bounding the Tree Cost

We can think of the algorithm as incurring a cost M per time unit for growing each component that does not contain s, the root vertex.

If $a(t)$ denotes the number of non-root components at time t, the total cost of growing the components is

$$M \int_0^\infty a(t)dt$$

where the integral is over the entire execution of the algorithm.

By standard arguments, the cost of the tree constructed is at most two times the growing cost.

Bounding the Tree Cost, con't

For each Steiner component C' we define a (time varying) set $\underline{Contrib(C')}$ of users that will be responsible for maintaining a steady flow of funding of at least $M/3$ per time unit from the time increments $f_j(t)$ of their cost shares.

Thus we have that, at any time t, the users are able to collect $Ma(t)$ revenue from their contributions at time $t/3$. Hence:

$$\int_0^\infty Ma(t)dt \leq \int_0^\infty \sum_j f_j(t/3)dt$$

$$= 3\int_0^\infty \sum_j f_j(t)dt = 3\sum_j \alpha_j$$

Since $M\int_0^\infty a(t)dt$ can pay for half of the tree, the buying costs of our solution does not exceed $6\sum_j \alpha_j$.

Main Result

Lemma. *The cost of the solution constructed is at most* $\sum_j (3\alpha'_j + 6\alpha_j)$.

Follows from previous slides.

Theorem. *The cost shares are cross-monotonic, competitive, and recover* 1/15 *fraction of the cost of the solution constructed*.

The 1/15 approximation factor follows from the fact that a feasible solution has a cost of at least: max $(1/2 \sum_{j \in J} \alpha_j, \sum_{j \in J} \alpha'_j)$, and the bound stated above.