CS 684: Algorithmic Game Theory **Scribe: Mikolaj Franaszczuk**
Instructor Eva Tardos **Friday, April 16, 2004**

Combinatorial Auctions

There is a set N of items, and players who want to bid on subsets of the items. It's combinatorial because a subset may contain more than one item, and the utility of two or more items together may be more then the sum of the utilities of the individual items.

In the general instance, the input is exponential in size, as each user would have to supply a bid for every possible subset. So instead, each user's input specifies both the subset to bid on, and the price they are willing to pay for it.

Formally, the bidders are "single minded bidders":
N is s a set of **k** items.
There are **n** users. Each user **I** has subset $\mathbf{S_i}$ in **N** and value $\mathbf{U_i}$ for receiving $\mathbf{S_i}$

Note that lying by asking for a superset may help a user, as if he wins he would get the set he wants anyway, as it's included in the superset.

Social Welfare

VCG for this problem to decide who gets what.

Define social welfare: a set of disjoint subsets that maximize total value.
Formally:
$$\max \left(\sum_{i \in I} Ui : \text{set } S_i \text{ for } i \in I \text{ disjoint} \right) = OPT$$

Payment bonus:
P1 = OPT if i is not in I

P2 = OPT – U_i if i ∈ I

This is not an easy optimization problem as it is just like set-packing, which is NP-complete. It can't even be approximated well, as will be shown later.

Heuristic Algorithm

Suppose you have a heuristic algorithm to optimize this (such as integer programming that is stopped after a fixed amount of computational time). Is it truthful?

Assume that users know what algorithm you are using to optimize. Then they will actually lie in order to help you, by giving you extra input! Their welfare is aligned with your optimization accuracy, so it benefits the users to help you optimize. So a heuristic approximation is not truthful.

A paper by Nison and Ronen describes a 2 phase heuristic algorithm:

1. Users announce **Si** and **Ui** to designer
2. Mechanism announces all requests, as well as the heuristics it is using
3. Users are allowed to offer alternates. **(<u>S</u>, <u>U</u>)**. Each user proposes alternate sets and utilities for all other users.
4. Run the heuristic algorithm on the original set **(S, U)**, and on all n alternates **(<u>S</u>, <u>U</u>)$_i$** for all i. Return the best result from all of those runs.

Approximations

Can we use an approximation algorithm that given an approximation bound? Yes, but they aren't very good.

Possible algorithms:

(1) Take a single **Si** with the maximum **Ui**. This is a k-approximation and n-approximation algorithm. The worst case is when a set of n individual users wanting individual items is better than some one user taking the entire set.This algorithm is truthful (user will report actual utilities).

(2) Sort by $\frac{Ui}{|Si|}$ and allocate greedily. This is also an n-approximation. A worst case example is when a user I wants some one item for a high price, but another user with slightly lower density wants the entire set. Since the first user will be serviced first, the 2nd user is denied the entire set and the total value is thus comparatively low.

(3) Sort by **Ui** and allocate greedily. This is also an n-approximation. A worst case example is when some one user wants an entire set, but all the other users want all of the items individually, with utilities lower by just epsilon.

(4) Sort by $\frac{Ui}{\sqrt{|Si|}}$ and allocate greedily.

Theorem: (4) is a $\sqrt{n}$ approximation.

Note: unless P=NP, no $n^{\frac{1}{2}-e}$ approximation is possible [due to Hastad]

Theorem: Any one of the strategies 2-4 above can be made truthful by proper payments.

***Proof*:**

First, note that it is never tempting to lie about Si. Declaring a larger set will still give you your original set, but will put you later in the sorting order for the algorithms. Declaring a smaller set is never good because even if you get it, you don't get the full set you originally wanted. But it can be tempting to lie about the utility. Increasing the utility will put you higher in the sorting order. Decreasing the utility could potentially lower your payment. So how should payments be set to avoid such lying?

Set **Pi = 0** if I is not selected

Set **Pi = min value Ui** such that algorithm includes i with this value.

***Claim*: Resulting procedure is truthful:**

If **i** $\in$ **I**, then user i already gets the lowest possible payment, so there is no incentive to decrease utility. There is no incentive to increase it either, as the user is already guaranteed to be in the set

If **i** is not in **I**, then raising **Ui** high enough to get the user in the set will exceed payment, so it won't be worth it to be in the set, so the user won't make this lie.