CS 886: Multiagent Systems
Assignment 2

Due February 20 . In this assignment you are expected to work individually. You may use any sources that you want, but you must cite them. My office hours are Tuesdays 2:30-3:30 in DC 2518. You can also email me if you have questions.

1. (10 points) Consider a private value auction of one good when bidders have quasi-linear utilities and know their own valuations. Prove that bidding truthfully is a dominant strategy in the Vickrey auction. (Prove this from first-principles; and not from the fact that the Vickrey auction is a special case of the VCG mechanism.)

2. (20 points)

   (a) Design an expected revenue maximizing auction mechanism (where participation is *ex-post* individually rational) for the following setting. You have one item to sell. You do not care about keeping it or getting rid of it. There are two bidders with quasi-linear utility functions. Bidder 1's valuation is drawn from uniform distribution on [0,1]. Bidder 2's valuation is drawn independently from uniform distribution on [1,4]. (Hint: Myerson auction)

   (b) What is your expected revenue in your mechanism?

   (c) What is your worst-case revenue in your mechanism?

   (d) Is your mechanism Pareto efficient? Justify your answer.

3. (30 points) (Thanks to K Leyton-Brown) Suppose you have some object that each of n agents desires, but which you do not value. Assume that each agent i values it at v_i with v_i's drawn independently and uniformly from some real line positive interval, say $(0,10^{100}]$. Although you do not desire the object and do not care about the actual values of the v_i's, you need to compute $\sqrt{v_i}$ for each i.

   Unfortunately, you face two problems. First, agents are not inclined to just reveal to you anything about their v_i's. Second, your computer is costly to operate. It costs you 1 unit to determine the greater of two values, 2 units to perform any basic arithmetic operation $(+, -, \times,)$, and anything more complicated (say $\sqrt{x}$) costs 20. The (accurate) current time of day can be observed without cost.

(a) How much does it cost to compute $\sqrt{v_i}$ for each i using a straightforward VCG mechanism? When computing cost, ignore the revenue that the auction will generate.

(b) Your answer above gives an upper bound on the cost of computing the square roots of the valuations. Design an incentive-compatible, dominant strategy ("strategy-proof") direct mechanism that will allow you to compute all $\sqrt{v_i}$ at minimal cost. Assume that the agents can do computations for free. Make sure that you specify all the components of the mechanism: players, actions, outcomes, mappings from actions to outcomes. Explain why your mechanism is strategy-proof. Specify the algorithms that you will use to implement those mappings. Give your mechanism's total computation cost (or an upper bound on it). You do not need to prove that your mechanism has minimal cost. (Hint: Think about the revelation principle.)

(c) In the previous part you were restricted to direct mechanisms. Show that an *indirect* (multistage) mechanism can achieve even lower cost.

4. (30 points) Recall that the OR* bidding language adds "dummy goods" to bids in a combinatorial auction in order to express complex bidding languages within the OR language. Consider a combinatorial auction with n items. For each valuation function below, explain how to encode it in the OR* language using as few bids as possible. Explain how many dummy items your encoding requires.

(a) $v(S) = k|S|$ for some constant k

(b) $v(S) = k|S|$ if $|S| \leq j$ for some j, otherwise $v(S) = 0$

(c) $v(S) = 1$ is $|S| \geq n/2$ and $v(S) = 0$ otherwise

(d) Goods come in three colours, red, blue and green; there are $n/3$ goods of each colour. $v(S) = k_R|S|$ if it contains only red items, $v(S) = k_G|S|$ if it has only green items, and $v(S) = k_B|S|$ if it has only blue items (for some constants k_R, k_G, k_B). $v(S) = 0$ otherwise.

(e) Goods come in pairs. $v(S) = |S|$ if it contains at most one item from every pair, and $v(S) = 0$ otherwise.