COS 445 - PSet 4

Due online Monday, April 13th at 11:59 pm

Problem 1: Noisy Optimizers aren't Good Enough (40 points)

For this problem, you should assume that all bidders' values for the item are non-negative. This problem will try to address the "robustness" of the second-price auction (or more generally, ideas used for VCG) to underlying optimization algorithms which are imperfect. Consider the following error-prone algorithm A for computing the argmaximum of a set $\{b_1, ..., b_n\}$ of numbers:

- If the *second-highest* number is exactly one less than the largest number, then output the index of the *second-highest* number (break ties lexicographically).[1]
- Otherwise, output the index of the largest number (break ties lexicographically).

Consider the following error-prone version of the second-price auction with n buyers and a single item:

- Accept bids $b_1, \ldots, b_n$, all of which are ≥ 0.
- Award the item to the bidder $A(b_1, ..., b_n)$. Note that if A were not error-prone, this would be the highest bidder.
- Charge the winning bidder $b_{A(\vec{b}_{-i};-2)}$.[2] To clarify, $A(\vec{b}_{-i}; -2)$ means "replace b_i with -2 and keep all other bids the same. Then run A." Put another way,: find the bidder j which A selects as the winner on input $(\vec{b}_{-i}; -2)$, and charge bidder i b_j. Note that if A were not error prone, j would be the highest bidder among those $\neq i$.

Part a (10 points)

Prove that for any given $\vec{b}_{-i}$ (list of bids submitted by all bidders except for i), there exists a price p such that no matter what bid bidder i makes, bidder i will either win the item and pay p, or not get the item (and pay nothing).

Part b (10 points)

Say that $v_i > v_j$ for all $j \neq i$ (i is the highest bidder). Prove that if all other bidders tell the truth (that is, bid v_j), bidder i's best response is a bid which wins the item (you do not need to specify exactly what that bid is).

[1] To be extra clear: if the third-highest number is exactly one less than the largest number, but the second-highest number is not, then the index of the highest number is output. If there are two numbers with the same highest value, then the second-highest number is equal to the highest number.

[2] The choice of -2 is made just to guarantee that $A(\vec{b}_{-i}; -2) \neq i$ when all $b_j \geq 0$.

Part c (10 points)

Prove that the error-prone second-price auction is not incentive compatible by providing (and analyzing) a vector of values $v_1, \ldots, v_n$ such that if everyone tells the truth, the second-highest bidder wins and pays strictly more than their value (pick an n and provide a single example. It is OK to use non-integer values, if desired).

Part d (10 points)

Prove that the error-prone second-price auction is not incentive compatible by providing (and analyzing) a vector of values $v_1, \ldots, v_n$ such that if everyone tells the truth, the highest bidder wins, but the second-highest bidder would have been strictly happier by lying about their value (pick an n and provide a single example. It is OK to use non-integer values, if desired).

Problem 2: Super Selfish Mining (40 points)

In this problem, you'll examine a variant of the selfish mining strategy. If it helps, you may think of the following as selfish mining done by a prophet who knows in advance who will be selected to mine in each round.[3] Throughout this problem, you are the attacker, and we will refer to you as m.

Imagine that you control an α fraction of the total computational power in the Bitcoin network, all other miners follow longest-chain and always tie-break against you, and you use the following mining strategy (which is different from lecture). We describe the strategy below in both text and math — it will be helpful to think of the order of operations during every time step as (a) a block is created by a randomly chosen miner, equal to m with probability α and $\neq m$ with probability $1-\alpha$, (b) all other miners broadcast all of their blocks, and then (c) you may broadcast any blocks you like.

Intuitively, your strategy is exploiting its knowledge of the future, and will *only* hide a block if you *know* that you will mine the next block and build a lead of two. Below, note that your decisions on what to do in step t *will depend* on whether or not you are mining the block in step $t+1$.

Super Selfish Mining:

- **Notation:** For every time step t,
  - Let M_t denote the identity of the miner who is selected to mine at time t. That is, if $M_t = m$, then the attacker is selected during step t. If $M_t \neq m$, then another miner is selected.
  - At all times t, let $h_m(t)$ denote the height of the highest block mined by you (computed after step (b), before step (c) when you are choosing what to broadcast), and let $h(t)$ denote the height of the highest block that has been broadcast publicly (again after step (b), before step (c)). Recall that the other miners know $h(t)$, but do not know $h_m(t)$ if some of your blocks are hidden.
- **Mining:** During every time step t, mine on top of the longest chain (among all blocks that were broadcast, or created by m), tie-breaking in favor of m (yourself).
- **Broadcasting when $M_t \neq m$:** During every time step t, if another miner broadcasts a block of height $h(t)$ during step (b) and:[4]
  1. $h_m(t) \leq h(t)$, announce your block of height $h(t)$, if one exists (if one does not exist, broadcast nothing).
  2. $h_m(t) > h(t)+1$, announce your block of height $h(t)$.
  3. $h_m(t) = h(t)+1$, announce your two blocks of height $h(t)$ and $h(t)+1$.
- **Broadcasting when $M_t = m$:** During every time step t, if $M_t = m$, decide whether to announce your new block according to the following rule:
  4. If $h_m(t) = h(t)+1$ **and** $M_{t+1} \neq m$, announce your block of height $h(t)+1$. That is, if you are mining this round, but *not* mining next round, *and* your new block only beats the public chain by one, broadcast your new block immediately.

[3] But if it does not help, you should solve the problem below, exactly as stated, and not focus on this. Note also that you may discover a smarter selfish mining strategy than the one defined below, but you should analyze exactly the strategy defined below, and not something more clever.

[4] Note that this will happen during any round where $M_t \neq m$.

5. If $h_m(t) = h(t) + 1$ **and** $M_{t+1} = m$, **do not announce** any blocks. That is, if you are mining this round *and next round*, hide your block.
6. If $M_t = m$ and $h_m(t) > h(t) + 1$, **do not announce** any blocks. That is, if you are mining this round and have a large secret lead, hide your block.

Part a (15 points)

Consider the case when miners are selected according to the following sequence (two, six, and seven are not m): $\langle M_1, M_2, M_3, M_4, M_5, M_6, M_7 \rangle = \langle m, \neq m, m, m, m, \neq m, \neq m \rangle$. Which of the six broadcasting cases is triggered during each round? You should present your answer in the form $\langle c_1, c_2, c_3, c_4, c_5, c_6, c_7 \rangle$, where each c_i is an integer from 1 to 6 denoting which case was triggered in round i, followed by a brief justification that each answer is computed correctly.

Part b (25 points)

Describe (or draw, if you prefer) a Markov chain (as a function of α) to analyze the expected reward achieved with the super selfish mining strategy defined above. You should also provide a brief explanation of why your analysis is correct. More specifically, you should provide:

1. A (possibly infinite) list of states.
2. For each pair of states, x and y, the probability of transitioning from state x to y, q_{xy}.[5]
3. For each pair of states, x and y, two values, H_{xy} and S_{xy} (think of H_{xy} as counting the number of blocks that the honest portion of the networks gets in the eventual longest chain when we transition from x to y, and S_{xy} as counting the number of blocks that the selfish miner gets in the eventual longest chain when we transition from x to y).[6]
4. If you find it convenient, you may create multiple "types" of transitions between two states, rather than rigorously add duplicate states. But there are solutions for which this is neither necessary nor convenient. You can ignore this bullet if you find it confusing and/or unhelpful.
5. Your brief justification should clearly prove that every block that is eventually in the longest chain is counted during exactly one transition, and also that only blocks which are in the longest chain are counted.[7]

Your solution should have the property that that if $\vec{p}$ denotes the stationary distribution of your Markov chain (that is, as time goes to infinity, for all x your Markov chain is at state x a p_x fraction of the time), the expected reward achieved by the super selfish mining strategy is:[8]

$$\frac{\sum_{x,y} p_x \cdot q_{xy} \cdot S_{xy}}{\sum_{x,y} p_x \cdot q_{xy} \cdot (S_{xy} + H_{xy})}.$$

You do not need to find the stationary probabilities of your Markov chain, nor compute the expected reward achieved by this strategy. You only need to describe the Markov chain

[5] If for many pairs, $q_{xy} = 0$, you may simply write "all other transition probabilities are zero" after you define the non-zero ones.

[6] Again, you must define all non-zero values, and can declare the rest to be zero.

[7] Your brief justification does not have to be set up to explicitly make both of these statements, but both of these statements should clearly and easily follow from your brief justification.

[8] Observe that if $\vec{p}$ is the stationary distribution, then $p_x \cdot q_{xy}$ is the fraction of time that we spend transitioning from state x to state y.

as detailed above, and briefly explain why the analysis is correct.

Hint: Refer to Lecture Notes and/or Precept Notes for an example of how to do this for non-super selfish mining.

Extra Credit: Proportionality for large n

Recall that extra credit is not directly added to your PSet scores, but will contribute to your participation. Some extra credits are **quite** challenging. We do not suggest attempting the extra credit problems for the sake of your grade, but only to engage deeper with the course material. If you are interested in pursuing an IW/thesis in CS theory, the extra credits will give you a taste of what that might be like.[9]

Let there be n players with normalized, additive values for a cake $[0, 1]$. Let also $\mathcal{A}$ denote the set of all partitions of cake to the n players. Let $\mathcal{P}$ denote the set of all *proportional* partitions of cake to the n players (that is, each player has value at least $1/n$ for their allocated cake).

For notation below, for an allocation $S := S_1 \sqcup \ldots \sqcup S_n$, let $V(S) := \sum_i V_i(S_i)$. Prove that for all n, and all valuations $V_1, \ldots, V_n$, $\max_{S \in \mathcal{A}}\{V(S)\} / \max_{S \in \mathcal{P}}\{V(S)\} = O(\sqrt{n})$. That is, the welfare of the best proportional allocation is at least an $O(\sqrt{n})$ factor of the best welfare without proportional constraints.

For all n, provide a list of valuations $V_1, \ldots, V_n$ such that $\max_{S \in \mathcal{A}}\{V(S)\} / \max_{S \in \mathcal{P}}\{V(S)\} = \Omega(\sqrt{n})$ (that is, prove that the previous bound is tight up to constant factors).

Hint: You will for sure want to use ideas from the algorithm we saw in class to find a proportional allocation.

Hint: Try to break it down into cases where not-that-many players contribute more than $1/\sqrt{n}$ to the total value, and those where many players contribute more than $1/\sqrt{n}$.

[9] Keep in mind, of course, that you will do an IW/thesis across an entire semester/year, and you are doing the extra credit in a week. Whether or not you make progress on the extra credit in a week is not the important part — it's whether or not you enjoy the process of tackling an extremely open-ended problem with little idea of where to get started.