CS 598RM: Algorithmic Game Theory, Fall 2019
HW 3 (due on Wednesday, 13th Nov at 11:59pm CST)

Instructions:

1. We will grade this assignment out of a total of 40 points.
2. Feel free to discuss with fellow students, but write your own answers. If you do discuss a problem with someone then write their names at the starting of the answer for that problem.
3. Please type your solutions if possible in Latex or doc whatever is suitable. We will upload submission instructions on the course webpage and Piazza.
4. Even if you are not able to solve a problem completely, do submit whatever you have. Partial proofs, high-level ideas, examples, and so on.
5. Except where otherwise noted, you may refer to lecture slides/notes, and to the references provided. You cannot refer to textbooks, handouts, or research papers that have not been listed. If you do use any approved sources, make sure you cite them appropriately, and make sure to write in your own words.
6. No late assignments will be accepted.
7. By AGT book we mean the following book: Algorithmic Game Theory (edited) by Nisan, Roughgarden, Tardos and Vazirani. Its free online version is available at Prof. Vijay V. Vazirani's webpage.

---

1. Consider a setting with a set M of m divisible goods and a set N of n players. Define an allocation $x \in \mathbb{R}^{n \times m}$ as an $n \times m$ matrix in which x_{ij}denotes the fraction of good j allocated to player i. Let $\mathcal{F} = \{x \mid x_{ij} \geq 0 \text{ and } \sum_i x_{ij} \leq 1\}$ denote the set of feasible allocations. Lastly, assume that each player i has a valuation function $v_i : \mathbb{R}^m_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $v_i(\emptyset) = 0$, i.e. each player i's valuation for the allocation $x' = c \cdot x$ satisfies $v_i(x') = c \cdot v_i(x)$ for any $c \geq 0$.

   We define Nash Fairness (NF) as follows. An allocation x^* is Nash fair if, for any other allocation x',the total proportional change in valuations is not positive, i.e.

$$\sum_{i \in N} \frac{v_i(x') - v_i(x^*)}{v_i(x^*)} \leq 0$$

   It is known that a NF allocation exists, and, in fact, it is the unique allocation that maximizes the Nash product $\prod_{i \in N} v_i(x)$; you may rely on this fact in your solution.

   The Partial Nash (PN) algorithm first computes the NF allocation x^*, and then assigns each player i a fraction of x^*_i that depends on the extent to which the presence of i inconveniences the other players (i.e., decreases the value of other players).

Algorithm 1: Partial Nash (PN) Algorithm

Compute the NF allocation x^* based on the reported bids.

For each player i, remove her and compute the NF allocation x^*_{-i} that would occur in her absence.

Allocate to each player i a fraction f_i of everything she receives according to x^* , where

$$f_i = \frac{\prod_{i' \neq i} v_{i'}(x^*)}{\prod_{i' \neq i} v_{i'}(x^*_{-i})}$$

   (a) (3 points) Show that the allocation produced by the PN algorithm is feasible.

   (b) (7 points) Prove that the PN algorithm is strategyproof; that is, no player can benefit by reporting untruthfully.

Solution.

2. (a) i. (3 points) Let $\mathcal{C}$ denote the set of quadratic cost functions, i.e. each $c \in \mathcal{C}$ is of the form $c(x) = ax^2 + bx + c$ with $a, b, c \geq 0$, and consider a Pigou like network where the cost functions on both edges are in $\mathcal{C}$. Show that the Pigou bound $\alpha(\mathcal{C})$ is $\frac{3\sqrt{3}}{3\sqrt{3}-2}$.

ii. **(Hard Bonus Question)** (5 points) Consider the following nonlinear variant of Pigous example.

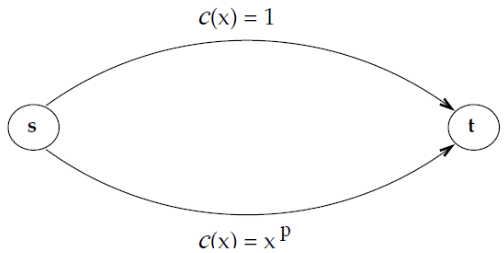

Figure 1: Nonlinear variant of Pigous example.

Show that the Pigou bound $\alpha(\mathcal{C})$ is

$$\frac{(p+1)\sqrt[p]{p+1}}{(p+1)\sqrt[p]{p+1}-p}$$

(b) Consider a combinatorial auction with n bidders and n items where each bidder i has a unit-demand valuation v_i. This means that $v_i(S) = \max_{j \in S} v_{i,j}$ for every subset S of items. We assume that $v_{i,j} > 0$ for all i, j.

In this auction, each bidder i submits one bid $b_{i,j}$ for each item j, and each item is sold separately using a second-price single-item auction. Assume that $b_{i,j} \in (0, v_{i,j})$ for all i, j. The utility of a bidder is her value for the items won, minus her total payment. For example, if bidder i has values v_{i1} and v_{i2} for two items, and wins both items when the second-highest bids are p_1 and p_2, then her utility is $\max\{v_{i1}, v_{i2}\} - (p_1 + p_2)$. Let $G = (A, B)$ be a bipartite graph where A is the set of bidders and B is the set of items.

i. (2 points) Show that every allocation π of items to bidders that maximizes the sum of valuations $\left(\sum_i v_{i,\pi(i)}\right)$ induces a perfect matching on G.

ii. (5 points) Show that the PoA of PNE in such a game can be at most 2.

Solution.

3. Consider n identical machines and m selfish jobs (the players). Each job j has a processing time p_j. Once jobs have chosen machines, the jobs on each machine are processed serially from shortest to longest. (You can assume that the p_j's are distinct.) For example, if jobs with processing times $1, 3$, and 5 are scheduled on a common machine, then they will complete at times $1, 4$, and 9, respectively. The following questions concern the game in which players choose machines in order to minimize their completion times. The objective function as a planner is to minimize the total completion time $\sum_{j=1}^{m} C_j$, where C_j is the completion time job j.

   (a) (3 points) Define the rank R_j of job j in a schedule as the number of jobs on j's machine with processing time at least p_j (including j itself). For example, if jobs with processing times $1, 3$, and 5 are scheduled on a common machine, then they have ranks $3, 2$, and 1, respectively.

   Prove that in these scheduling games, the objective function value of an outcome can also be written as $\sum_{j=1}^{m} p_j R_j$.

   (b) (3 points) Prove that the following algorithm produces an optimal outcome: (i) sort the jobs from largest to smallest; (ii) for $i = 1, 2, \ldots, m$, assign the i^{th} job in this ordering to machine i mod n (where machine 0 means machine n).

   (c) (4 points) Prove that for every such scheduling game, the expected objective function value of every coarse correlated equilibrium is at most twice that of an optimal outcome.

   Hint: *The (λ, μ)-smoothness condition (see notes provided) was required for all pairs $\boldsymbol{s}^*, \boldsymbol{s}$ of outcomes. Weaken the definition so that this condition only needs to hold for some optimal outcome $\boldsymbol{s}^*$ and all outcomes $\boldsymbol{s}$. Observe that PoA of coarse correlated equilibria remains at most $\frac{\lambda}{1-\mu}$ assuming only this weaker condition (with the same proof as before). Prove that this scheduling game satisfies this weaker condition for $\lambda = 2$ and $\mu = 0$.*

Solution.

4. (a) (3 points) Give an example of a game with 2 players that admits a PNE, but the best-response dynamics cycles.

   (b) This problem studies a scenario with n agents, where agent i has a positive weight $w_i > 0$. There are m identical machines. Each agent chooses a machine, and wants to minimize the load of her machine, defined as the sum of the weights of the agents who choose it. A pure Nash equilibrium in this game is an assignment of agents to machines so that no agent can unilaterally switch machines and decrease the load she experiences.

   Consider the following restriction of best-response dynamics:

   Algorithm 2: Maximum Weight Best-Response Dynamics

   While the current outcome $\boldsymbol{s}$ is not a PNE:
   among all agents with a beneficial deviation, let i denote an agent with the largest weight w_i and s_i a best response to $\boldsymbol{s}_i$
   update the outcome to $(s_i, \boldsymbol{s}_i)$

   i. (2 points) Show that, starting from the outcome $\boldsymbol{s}_0$ where no agent has selected any machines (all machines have load 0), the Maximum Weight Best-Response Dynamics converges to a PNE in exactly n iterations.

   ii. (5 points) Show that, starting from any outcome $\boldsymbol{s}$, the Maximum Weight Best-Response Dynamics converges to a PNE in at most n iterations.

Solution.