CO759: Algorithmic Game Theory — Spring 2015

Instructor: Chaitanya Swamy

Assignment 2

Due: By Aug 14, 2015

You may use anything proved in class directly. I will maintain a FAQ about the assignment on the course webpage. *Acknowledge all collaborators and any external sources of help or reference.*

There are currently 7 questions in the assignment; 2 more will be added next week. You may answer *any 5* questions. All questions carry equal weightage.

Q1: In this problem, we consider *multi-unit combinatorial auctions* (MUCAs), which recall are CAs where the m items are all *identical*. Thus, each player i's valuation function v_i can be represented as a nondecreasing function $v_i : \{0, 1, \ldots, m\} \mapsto \mathbb{R}_+$ where $v_i(x)$ specifies the value that i receives when he is allotted x items. The SWM problem of finding x_i's in $\{0, 1 \ldots, m\}$ with $\sum_i x_i \leq m$ that maximize $\sum_i v_i(x_i)$ is solvable in time polynomial in m and n. So if the input is specified by listing $v_i(x)$ for each x separately, then the running time is polynomial in the input-size, and one can implement the VCG mechanism efficiently. We are interested in settings where the v_i's are specified more succinctly and the input-length is polynomial in $\log m$; for example, if each $v_i(.)$ is piecewise linear with a constant number of breakpoints then one only needs to list $(x, v_i(x))$ for each breakpoint x. The SWM problem is often *NP*-hard in such settings.

Let $A = \{(x_1, \ldots, x_n) : x_i \in \{0, \ldots, m\},\ \sum_i x_i \leq m\}$ denote the set of all allocations, and let $A' = \{(x_1, \ldots, x_n) : x_i \in \{0, \ldots, m\},\ \sum_i x_i \leq m,\ x_i = m$ or x_i is a multiple of $\lceil \frac{m}{n^2} \rceil\}$. Consider the function f that returns the allocation in A' that maximizes the social welfare among all allocations in A'.

(a) Prove that f is a 0.5-approximation algorithm, i.e., show that $\max_{(x_1,\ldots,x_n)\in A'} \sum_i v_i(x_i) \geq 0.5 \cdot \max_{(x_1,\ldots,x_n)\in A} \sum_i v_i(x_i)$.

(b) Briefly argue why f is implementable.

(c) Assuming that for every i and every integer $x \in [0, m]$ one can compute $v_i(x)$ efficiently, show that f can be computed efficiently.

(**Hint:** Use dynamic-programming.)

Q2: Recall the load balancing game from class: there are m machines and n jobs. Each job j is a player: it has a certain processing time p_j and its strategy is to choose a machine to get assigned to. Given an assignment of jobs to machines, the load L_i of machine i is the total processing time of the jobs assigned to it, and the cost incurred by a job j is the load of the machine to which it is assigned. In this question, we address the issue of how quickly the natural heuristic of letting players make improving-moves converges to a Nash equilibrium. (You might want to read the proof of Theorem 20.6 in the book before attempting this question.)

Given an assignment $\{i(j)\}$, we say that a job j is *unsatisfied* if it can move to some machine (other than $i(j)$) and reduce its cost. A *best-response move* of job j is a move that minimizes its cost (given the current assignment).

(a) Consider the round-robin policy, where jobs are considered in some arbitrary order and if a job is unsatisfied it makes a best-response move. Prove that this policy leads to a (pure) Nash equilibrium in at most n^2 steps. (Each iteration counts as a step regardless of whether the job considered is unsatisfied or not.)

(b) Now consider the following randomized policy: in each step, pick one of the n jobs at random and if it is unsatisfied make a best-response move. Prove that with probability at least $1 - \frac{1}{n}$, this policy leads to a Nash equilibrium in $O(n^2 \log n)$ steps.

Q3: Consider a nonatomic routing game instance $\mathcal{I} = \big(G, \{\ell_e(.)\}, \{(s_i, t_i, r_i)\}_{i=1}^k\big)$ where G is the underlying directed graph, the $\ell_e(.)$'s are the continuous nondecreasing latency functions on the edges, and r_i units of flow have to be routed from the source s_i to sink t_i for each $i = 1, \ldots, k$. Let $\mathcal{P}_i$ denote the set of all s_i-t_i paths. Recall that $C(f) = \sum_e f_e \ell_e(f_e) = \sum_i \sum_{P \in \mathcal{P}_i} f_P \ell_P(f)$ denotes the total cost of a feasible flow f. In class, we proved that $C(f^{NE}) \leq C(f^*)$, where f^{NE} is a Nash flow for $\mathcal{I}$ and f^* is an optimal flow for the instance $2\mathcal{I} = \big(G, \{\ell_e(.)\}, \{(s_i, t_i, 2r_i)\}\big)$.

Given a class $\mathcal{L}$ of latency functions, recall that $\alpha(\mathcal{L}) = \sup_{\ell \in \mathcal{L}} \alpha(\ell)$ and $\alpha(\ell) = \sup_{a,b \geq 0} \frac{b \cdot \ell(b)}{a \cdot \ell(a) + (b-a) \cdot \ell(b)}$. Define $\rho = \rho(\mathcal{L}) = 2 - \frac{1}{\alpha(\mathcal{L})} < 2$. Show that if all the latency functions belong to class $\mathcal{L}$, then the above bound can be improved to $C(f^{NE}) \leq C(\tilde{f})$, where $\tilde{f}$ is an optimal flow for the instance $\rho\mathcal{I} = \big(G, \{\ell_e(.)\}, \{(s_i, t_i, \rho r_i)\}\big)$.

Q4: In this question, we explore applications and generalizations of the notion of smooth games.

(a) Prove that for atomic routing games with latency functions that are polynomials of degree p with nonnegative coefficients, the PoA with respect to *coarse-correlated equilibria* is at most $p^{O(p)}$.

(**Hint:** Show that $\alpha(\beta + 1)^p \leq \big(h(p) \cdot \alpha^{p+1} + \beta^{p+1}\big)/2$, where $h(p) = p^{O(p)}$ is an increasing function of p, and use the framework of smooth games.)

(b) Let $\mathcal{G} = \big(n; S_1, \ldots, S_n; \{C_i : S \mapsto \mathbb{R}_+\}\big)$ be a game on n players, where each player i has strategy-set S_i, and incurs cost $C_i(s)$ under each strategy profile $s \in S := \prod_i S_i$. Define $C(s) := \sum_{i=1}^n C_i(s)$, and let $OPT = \min_{s \in S} C(s)$. Recall that in class, we defined $\mathcal{G}$ to be weakly (λ, μ)-smooth if there exists $s^* \in S$ with $C(s^*) = OPT$ such that

$$\sum_{i=1}^n C_i(s_i^*, s_{-i}) \leq \lambda \cdot C(s^*) + \mu \cdot C(s) \qquad \text{for all } s \in S. \tag{1}$$

Notice that in (1), the strategy profile s^*, and hence, the deviation s_i^* of each player i, may not depend on s. We now consider a relaxation where the deviation s_i^* may *partly* depend on s and explore the consequences of this relaxation. Define $\mathcal{G}$ to be *adaptively* (λ, μ)*-smooth* if there exists a function $f_i : S_i \mapsto S_i$ for every player i such that

$$\sum_{i=1}^n C_i\big(f(s_i), s_{-i}\big) \leq \lambda \cdot OPT + \mu \cdot C(s) \qquad \text{for all } s \in S. \tag{2}$$

Prove that if $\mathcal{G}$ is adaptively (λ, μ)-smooth, then the PoA with respect to *correlated equilibria* is at most $\frac{\lambda}{1-\mu}$. (This PoA bound need not however hold for coarse-correlated equilibria.)

Q5: In this question, we consider the PoA (of pure NE) of atomic and nonatomic routing games with respect to the *maximum player-cost objective* that we looked into in the load balancing game. We are given a directed graph G with nondecreasing latency functions $\{\ell_e(.)\}$ on the edges. We will restrict our attention to instances where all traffic has to be routed from a common source s to a common sink t. Given an s-t flow $f = (f_P)$ that sends flow $f_P \geq 0$ on the s-t path P, let $L(f) = \max_{P:f_P>0} \ell_P(f)$ (where $\ell_P(f) = \sum_{e \in P} \ell_e(f_e)$) denote the cost of the flow according to the maximum-cost objective.

(a) Consider the atomic routing game where there are k players, each having source s and sink t. Let $\{P_i\}_{i=1}^k$ be a (arbitrary) Nash equilibrium with associated flow-vector f, so that $L(f) = \max_i \ell_{P_i}(f)$. Show that the PoA for the maximum-cost objective $L(.)$ with linear latency functions is at most $\frac{5}{2}$.

(b) Now consider the nonatomic game where k units of flow have to be sent from s to t and any fractional amount of flow can be routed along any s-t path. Let α be the PoA for the *total-cost objective* $C(f) = \sum_P f_P \ell_P(f) = \sum_e f_e \ell_e(f_e)$ with the given latency functions. Show that the PoA for the maximum-cost objective $L(.)$ is at most α.

Q6: We explore a notion of fairness for flows in this question (which is related to the maximum player-cost objective). Let $\mathcal{I} = \big(G, \{\ell_e(.)\}, \{(s_i, t_i, r_i)\}\big)$ be a nonatomic routing instance, where the latency functions $\ell_e(.)$ are continuous, nondecreasing, and further assume that the function $x\ell_e(x)$ is convex for all e. Consider the total-cost objective $C(f) = \sum_e f_e \ell_e(f_e) = \sum_P f_P \ell_P(f)$.

(a) Let f^{OPT} be an optimal flow for $\mathcal{I}$. Show that if all the latency functions are linear, then $2f^{OPT}$ is a Nash flow for the instance $2\mathcal{I} = \big(G, \{\ell_e(.)\}, \{(s_i, t_i, 2r_i)\}\big)$.

(**Hint:** Recall the fact stated in class that f^{OPT} may be viewed as the Nash flow with respect to a modified set of latency functions.)

(b) Notice that a Nash flow, by definition, has a fairness property associated with it, namely, that any two players routing flow between the same s_i-t_i pair experience the same total delay. (That is, for every s_i-t_i pair, every flow-carrying path P has the same total delay $\ell_P(f)$.) An optimal flow need not (and most likely, will not) be fair in this sense. Here we quantify the unfairness of optimal flows. Given an arbitrary feasible flow f for $\mathcal{I}$ along with a path-decomposition (f_P), we define the *unfairness* of the flow f to be $\max_i \big[\big(\max_{P \in \mathcal{P}_i: f_P>0} \ell_P(f)\big) / \big(\min_{P \in \mathcal{P}_i: f_P>0} \ell_P(f)\big)\big]$, where $\mathcal{P}_i$ is the set of all s_i-t_i paths. (This definition assumes that only paths carrying flow have a bearing on the discontent of users, which is plausible if one assumes that users "learn" about the existence of a better path only by seeing other users on that path.) Prove that with linear latency functions, the unfairness of any optimal flow under any path-decomposition is at most 2. (As mentioned earlier, the unfairness of any Nash flow is 1.)

Q7: Recall that for the atomic routing game, we used the (exact) potential function, $\Phi(f) = \sum_e \sum_{x=1}^{f_e} \ell_e(x)$ to prove the existence of a pure NE. Here we show how the potential function can also be useful in finding an *approximate* Nash equilibrium quickly.

Consider the atomic routing game on a directed graph G with nondecreasing latency functions $\{\ell_e(.)\}$, and k players, all having a common source s and common sink t. Consider the total-cost

objective $C(f) = \sum_e f_e \ell_e(f_e) = \sum_i \ell_{P_i}(f)$, so Φ satisfies $\Phi(f) \le C(f)$ for all f. Also, let $\alpha \ge 1$ be such that $\ell_e(x+1) \le \alpha \cdot \ell_e(x)$ for all e and all $x \in \{1, \ldots, k-1\}$. Let $\mathcal{P}$ denote the set of all s-t paths. Given a flow-vector f corresponding to a path-selection $\{P_i\}_{i=1}^k$, where each P_i is an s-t path, define the *neighborhood* of f to be

$$N(f) \;:=\; \big\{g : \exists i \in \{1, \ldots, k\} \text{ and } Q \in \mathcal{P} \text{ s.t. } g \text{ is the flow-vector associated with } (Q, P_{-i})\big\}.$$

Say that $\big(\{P_i\}, f\big)$ is an *ϵ-approximate local optimum* of Φ, if for all $g \in N(f)$, we have $\Phi(f) - \Phi(g) \le \epsilon \cdot \Phi(f)$. Define $\big(\{P_i\}, f\big)$ to be an *ϵ-Nash equilibrium* if for all i, for all $Q \in \mathcal{P}$, we have $\ell_{P_i}(f) - \ell_Q(g) \le \epsilon \cdot \ell_{P_i}(f)$, where $g \in N(f)$ is the flow-vector that results when i deviates from P_i to Q.

(a) Show that if $\big(\{P_i\}, f\big)$ is an ϵ-approximate local optimum of Φ, then $\big(\{P_i\}, f\big)$ is a δ-NE, where $\delta = \frac{\epsilon k \alpha}{1 - k\epsilon}$.

(**Hint:** Try to lower bound $\ell_{P_i}(f)$ in terms of $\Phi(f)$ for every $i = 1, \ldots, k$.)

(b) Prove the following "contrapositive" of part (a). Given $\big(\{P_i\}, f\big)$, let player i^* and $Q^* \in \mathcal{P}$ be such that among all the possible moves where a single player switches to an alternative strategy, the move where i^* switches over to Q^* yields the maximum reduction in the deviating player's cost. In other words, letting g^* denote the flow-vector in $N(f)$ obtained when i^* switches over to Q^*, we have $\ell_{P_{i^*}}(f) - \ell_{Q^*}(g^*) = \Phi(f) - \Phi(g^*) = \max_{g \in N(f)} \big(\Phi(f) - \Phi(g)\big)$. Show that if $\ell_{P_{i^*}}(f) - \ell_{Q^*}(g^*) \ge \epsilon \cdot \ell_{P_{i^*}}(f)$ then $\Phi(f) - \Phi(g^*) \ge \frac{\epsilon}{k(\alpha+1)} \cdot \Phi(f)$.

Part (a) shows that *any* algorithm for computing an approximate local-optimum of Φ can be used to compute an approximate Nash equilibrium. Using part (b), one can argue that an approximate local optimum may be computed by choosing improving moves suitably. Together, parts (a) and (b) yield a simple, efficient, "improving-moves" algorithm for computing a δ-NE. Suppose that $\Phi(f) \ge 1$ for all flow-vectors f resulting from a valid path-selection. Let C be some trivial upper bound on the maximum cost incurred by a player in a NE. For example, one can take $C = \min_{P \in \mathcal{P}} \sum_{e \in P} \ell_e(k)$. Assume that $\delta \le 1$ without loss of generality. We set $\epsilon = \frac{\delta}{2k\alpha}$, so $\frac{\epsilon k \alpha}{1 - k\epsilon} \le \delta$, and $\varepsilon = \frac{\epsilon}{k(\alpha+1)}$. We start from any state with $\Phi(.)$ value at most kC. At each step, given the current flow-vector f, letting i^*, Q^*, g^* be as defined in part (b), we make the improving move where i^* switches to path Q^* if the reduction $\ell_{P_{i^*}}(f) - \ell_{Q^*}(g^*)$ is at least $\epsilon \cdot \ell_{P_{i^*}}(f)$. We terminate when this no longer holds. Part (b) shows that each such improving move decreases the potential by at least a $(1 - \varepsilon)$-factor. So since the final potential is at least 1, the number of steps to termination is at most $\ln(kC) / \ln\big((1-\varepsilon)^{-1}\big) \le \frac{8k^2\alpha^2}{\delta} \cdot \ln(kC)$. At termination, we have

$$\max_{g \in N(f)} \big(\Phi(f) - \Phi(g)\big) = \ell_{P_{i^*}}(f) - \ell_{Q^*}(g^*) \le \epsilon \cdot \ell_{P_{i^*}}(f) \le \epsilon \cdot \Phi(f).$$

So f is an ϵ-approximate local optimum of Φ, and part (a) shows that hence, f is a δ-NE.

Q8: In this question, we devise algorithms for computing ϵ-NE for bimatrix games. Let (R, C) be a bimatrix game, where $R, C \in [0,1]^{m \times n}$ denote as usual the payoff matrices of the row and column players respectively. Recall that for an integer k, $[k]$ denotes the set $\{1, \ldots, k\}$.

(a) Recall the following algorithm mentioned in class for computing a 0.5-NE. Pick some pure strategy x for the row player. Let y be the best response of the column player to x (which we may assume is a pure strategy), and x' be the row-player's best response to y. Prove that the mixed-strategy profile $\big(0.5(x + x'), y\big)$ is a 0.5-NE of (R, C).

(b) We now devise an improved algorithm that returns an α-NE, where $\alpha = \frac{3-\sqrt{5}}{2} \approx 0.382$. (Note that α is the unique value in $[0,1]$ satisfying $\alpha = \frac{1-\alpha}{2-\alpha}$.) The idea is to consider the *zero-sum game* (A, B), where $A = R - C$ and $B = C - R$ denote the row- and column- player's payoffs. Zero-sum games enjoy various nice properties, one of them being that a Nash equilibrium for such a game can be computed efficiently. Consider the following algorithm.

A1. Compute a NE (x^*, y^*) of the zero-sum game (A, B).

A2. Compute $g_R = \max_{i\in[m]}(Ry^*)_i - x^{*T}Ry^*$ and let s_R be a row-player (pure) strategy that attains g_R, that is, $g_R = (Ry^*)_{s_R} - x^{*T}Ry^*$. Similarly, compute $g_C = \max_{j\in[n]}(x^{*T}C)_j - x^{*T}Cy^*$ and let s_C be a column-player (pure) strategy that attains g_C. (Observe that s_R and s_C are best responses of the row- and column- player's to y^* and x^* respectively.)

A3. If $g_R, g_C \leq \alpha$, return (x^*, y^*).

A4. Otherwise, do the following.

(i) If $g_R \geq g_C$ (and so $g_R > \alpha$), let b_C be the column-player's best response to s_R; return $\big(s_R, (1-\delta_R)y^* + \delta_R b_C\big)$, where $\delta_R = \frac{1-g_R}{2-g_R}$.

(ii) If $g_C > g_R$, let b_R be the row-player's best response to s_C; return $\big((1-\delta_C)x^* + \delta_C b_R, s_C\big)$, where $\delta_C = \frac{1-g_C}{2-g_C}$.

Prove that the above algorithm returns an α-NE.

(**Hint:** Consider for instance the strategy profile (x, y) returned in step 3(i) (the other case is symmetric). It is easy to see that the maximum gain, $\max_{i\in[m]}(Ry)_i - x^T Ry$, of the row player is at most δ_R. Observe that (x^*, y^*) being a NE of (A, B) implies that, fixing the column-player's strategy to be y^*, the row-player's change in payoff when he deviates to a strategy x' is at most the column-player's change in payoff under this deviation. Use this to show that the column-player's maximum gain is at most $(1-\delta_R)(1-g_R) = \delta_R$.)

(c) We now use the zero-sum game in part (b) to compute a 0.5-well-supported NE (WSNE) when $R_{ij}, C_{ij} \in \{0, 1\}$ for all $i \in [m], j \in [n]$; such games are sometimes called *win-lose games*. Prove that the following algorithm returns a 0.5-WSNE: if there exists $i \in [m], j \in [n]$ such that $R_{ij} = 1 = C_{ij}$, then return the (pure-) strategy profile (i, j); otherwise return a NE (x^*, y^*) of the zero-sum game $(R - C, C - R)$.