CS364A: Take-Home Final

Due to Peerapong in Gates 460 by 6:30 PM on Thursday, December 11, 2008

Instructions: Same as the problem sets.

Problem 21

(a) (6 points) *Algorithmic Game Theory*, Exercise 1.2.

(b) (9 points) *Algorithmic Game Theory*, Exercise 1.4.

Problem 22

Let A denote the payoff matrix of a two-player zero-sum game, with A_{ij} denoting the integral payoff from the column player to the row player in the outcome (i, j). Let R and C denote $\max_x(\min_y x^T Ay)$ and $\min_y(\max_x x^T Ay)$, respectively. (Here x and y denote mixed row and column strategies, respectively.)

(a) (4 points) At one point in lecture the instructor incorrectly defined a minimax pair as mixed strategies $\hat{x}, \hat{y}$ with the property that $R = \hat{x}^T A\hat{y} = C$. Show by example that such a pair $\hat{x}, \hat{y}$ need not be a mixed-strategy Nash equilibrium of the game.

(b) (6 points) The correct definition of a minimax pair $\hat{x}, \hat{y}$ is that $\hat{x} \in \arg\max_x(\min_y x^T Ay)$ and $\hat{y} \in \arg\min_y(\max_x x^T Ay)$. Prove that the mixed-strategy Nash equilibria (of a two-player, zero-sum game) are precisely the minimax pairs.

(c) (5 points) Prove that such a minimax pair (and hence a mixed-strategy Nash equilibrium) can be computed in polynomial time.

[Hint: you should assume and use the fact that linear programming can be solved in polynomial time.]

(d) (5 points) Can you run your same algorithm in (c) for a two-player, *non*-zero-sum game? If so, what is it computing, exactly?

Problem 23

Consider a two-player, non-zero sum game, specified by payoff matrices A and B, in which both players have n strategies. Assume that all payoffs are rational numbers between 0 and 1.

(a) (4 points) The *support* of a mixed strategy is the set of pure strategies that are played with strictly positive probability. Suppose you were promised that there was a mixed-strategy Nash equilibrium of (A, B) with row and column supports S and T, respectively. Show how you can then recover such an equilibrium in polynomial time.

[Hint: again, feel free to use linear programming.]

(b) (3 points) Give an $O(2^n)$-time algorithm to compute an exact mixed-strategy Nash equilibrium of a two-player game.

(c) (9 points) Consider a mixed-strategy Nash equilibrium (x^*, y^*) of (A, B). Suppose we draw $K = (c \log n)/\epsilon^2$ independent samples $(r_1, c_1), \ldots, (r_K, c_K)$ from the product distribution $x^* \times y^*$, where $\epsilon > 0$ is a parameter and $c > 0$ is a sufficiently large constant. Form the corresponding empirical distributions $\hat{x}$ and $\hat{y}$, where components are the frequencies of play in the sequence. Prove that, with high probability, $(\hat{x}, \hat{y})$ is an approximate Nash equilibrium in the sense that $\hat{x}^T A \hat{y} \geq x^T A \hat{y} - \epsilon$ for all row mixed strategies x and $\hat{x}^T B \hat{y} \geq \hat{x}^T B y - \epsilon$ for all column mixed strategies y.

[Hint: if you don't know about them, read up about Chernoff Bounds (e.g., in Motwani/Raghavan's *Randomized Algorithms*).]

(d) (4 points) Use (a) and (c) to give an $O(n^{(\log n)/\epsilon^2})$-time algorithm for computing an ϵ-approximate mixed-strategy Nash equilibrium of a two-player game.

Problem 24

We continue the study of algorithms for approximate Nash equilibria in bimatrix games, in the sense of the previous problem; recall especially the definition of approximate equilibria from Problem 23(c).

(a) (5 points) Prove that there is a $\frac{1}{2}$-approximate Nash equilibrium in which one player plays a pure strategy and the other randomizes uniformly between two strategies.

(b) (10 points) Consider the following idea for improving on (a). Suppose we form the matrix $A - B$, view it as a zero-sum game, and solve for an exact mixed-strategy Nash equilibrium (x^*, y^*). (By Problem 22(c), this can be done in polynomial time.) Perhaps (x^*, y^*) is an α-approximate Nash equilibrium for (A, B) for a satisfactory value of α; if not, we generalize (a) and have one player play a pure strategy while the other randomizes (not necessarily uniformly) between a pure strategy and its optimal strategy for $A - B$.

Can you use these ideas (with appropriate parameter choices) to improve upon the result in (a)? What is the minimum value of ϵ for which your algorithm computes an ϵ-approximate Nash equilibrium in polynomial time?

(c) (up to 10 extra credit points) Generalize the algorithm in (b) to compute ϵ-approximate Nash equilibria, in polynomial time, for even smaller values of ϵ. What is the smallest value you can achieve?

Problem 25

Consider the following variant of the online learning problem discussed in class. At each time step $t = 1, 2, \ldots, T$, n different "experts" give you advice as to some binary event (in the form of a 0 or 1). You have to make your own guess as to the event's realization (e.g., by following one of the expert's advice) and afterwards you are told whether you were right or wrong. You are promised that at least one expert is *omniscient* in that, at each time step, it always predicts the actual realization of the event. Your algorithm will be measured by its worst-case number of incorrect predictions, over all possible sequences of expert predictions and event realizations.

(a) (5 points) Prove that the minimum number of mistakes that a deterministic algorithm can make (in the worst case) is precisely $\log_2 n$.

(b) (10 points) Prove that the minimum expected number of mistakes that a randomized algorithm can make (in the worst case) is precisely $\frac{1}{2} \log_2 n$.

[Hint: use the distribution induced by the advice of the remaining potentially omniscient experts.]