UPenn NETS 412: Algorithmic Game Theory
Midterm Practice Problems

Instructor: Bo Waggoner

Problem 1

Here we'll analyze a game where each player has infinitely many choices of actions.

Alice and Bob are working on a project together. They both value their free-time, which they give up for working on the project. Let x_a represent the fraction of the day (expressed as a real value in $[0,1]$) Alice spends on the project and x_b the fraction of the day Bob spends. The value of the outcome of the project is given by $4x_1x_2$, and Alice and Bob split this quantity equally. The payoff to each player is the value of the outcome of the project minus the effort that player put it.

Part a Write Alice's payoff as a function of x_a and x_b.

$$u_a = \cdots\cdots\cdots$$

Part b Write Bob's payoff as a function of x_a and x_b.

Part c If Bob puts in effort $x_b = .75$, what is Alice's best-response?

Part d Now, if Alice is putting in effort $x_a = 1$, what is Bob's best-response?

Part e Why is $x_a = x_b = 1$ a Nash equilibrium of this game?

Part f Now, if Alice is putting in effort $x_a = .25$, what is Bob's best-response?

Part g What is Alice's best-response to Bob's choice of $x_b = 0$?

Part h Why is $x_a = x_b = 0$ a Nash equilibrium?

Part i What value of x_b makes Alice indifferent between all of her possible choices of x_a? Hint: Alice is indifferent between all of her options when her payoff is the same regardless of her choice of x_a.

Part j Why is $x_a = x_b = .5$ a Nash equilibrium?

Problem 2

Traffic is a congestion game played by n players, all starting from a source node S and traveling along routes represented by directed edges to a common destination node T. Each player's action set consists of all paths from A to T.

The delay along any edge (u, v) is equal to the number of players who chose to travel that edge, which we will write $n(u, v)$. Each player i's cost is equal to the sum of delays along all edges in her path.

For parts a and b, consider the following graph. Here, $n(S, A)$ refers to the number of players who travel along edge (S, A), and so on.

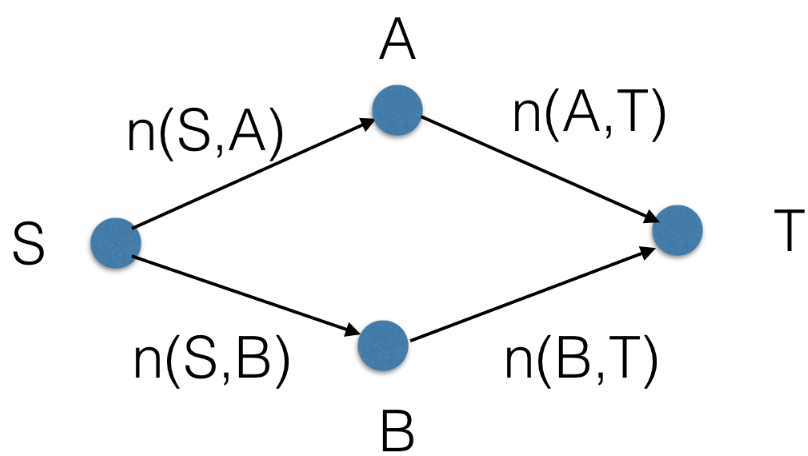

Figure 1: Graph for parts a and b

Part a Suppose n is even. Find a pure strategy Nash equilibrium, and prove that it is an equilibrium.

Part b Suppose n is odd. Find a pure strategy Nash equilibrium, and prove that it is an equilibrium.

Part c For this part, consider the following graph. Outline the argument that this game has a pure strategy Nash equilbrium. (You do not have to find it!) You can assume in your argument that this is a congestion game (it is actually a slight generalization).

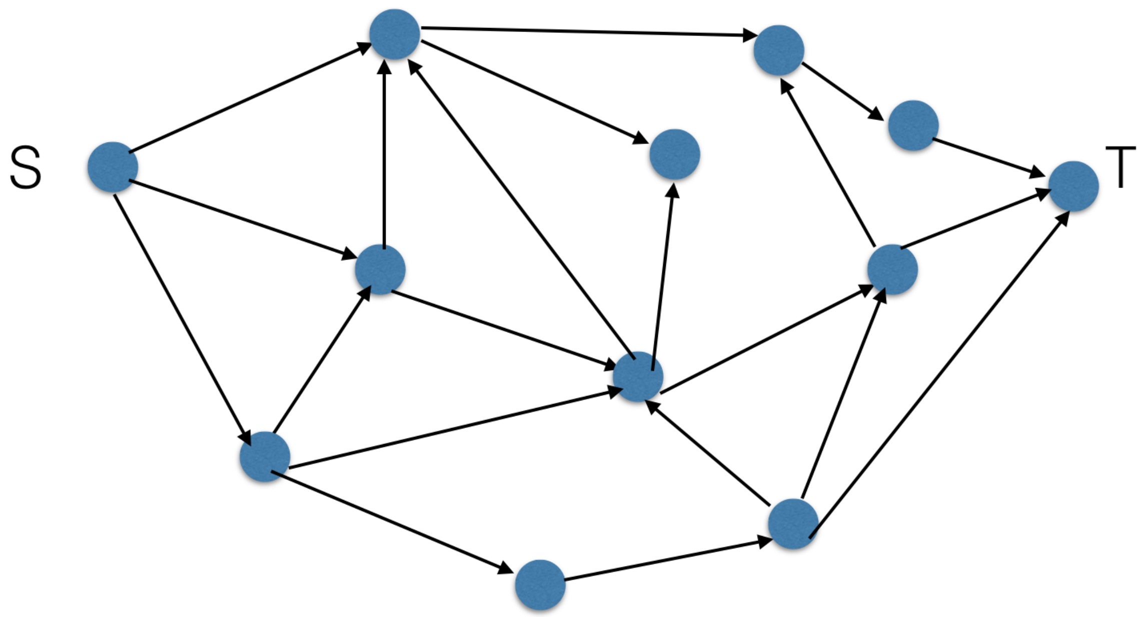

Figure 2: Graph for part c

Problem 3

In class, we proved that the polynomial weights algorithm achieves "no regret" – that is, for arbitrary sequences of losses, it guarantees that the difference between the average loss achieved by the algorithm, and the average loss achieved by the best expert in hindsight is $o(1)$ – tending to zero as $T \to \infty$. Recall that the polynomial weights algorithm is randomized. Here we show that no deterministic algorithm can obtain the same guarantee.

Part a Consider the algorithm "Follow the Leader" that always picks the expert that has the lowest cumulative loss so far – i.e. at day j it picks expert k such that:

$$k = \arg\min_i \sum_{t=1}^{j-1} \ell_i^t.$$

(Suppose for concreteness that if there is a tie, the algorithm picks the min-loss expert with the smallest index). Show that Follow the Leader is not a no-regret algorithm. i.e. exhibit a sequence of losses such that for all T, the regret of the algorithm is $\Omega(T)$.

Part b Prove that no *deterministic* experts algorithm can achieve $o(1)$ regret – i.e. that randomization is necessary to achieve a guarantee like that of the polynomial weights algorithm.

Hint: Consider any fixed deterministic algorithm, and then place yourself in the role of an adversary who is trying to foil it. Can you, knowing which expert the algorithm is going to pick next, design a sequence of losses so that after T rounds, the best expert always has cumulative loss that is lower than the algorithm's loss by at least $T/2$?