

COS 445 - PSet 3

Due online Monday, March 24th at 11:59 pm.

Instructions:

- Some problems will be marked as *no collaboration* problems. This is to make sure you have experience solving a problem start-to-finish by yourself in preparation for the midterms/final. You cannot collaborate with other students or the Internet for these problems (you may still use the referenced sources and lecture notes). You may ask the course staff clarifying questions, but we will generally not give hints.
- Submit your solution to each problem as a **separate PDF** to codePost. Please make sure you're uploading the correct PDFs to the correct locations!¹ If you collaborated with other students, or consulted an outside resource, submit a (very brief) collaboration statement as well. Please anonymize your submission, although there are no repercussions if you forget.
- The [cheatsheet](#) gives problem solving tips, and tips for a “good proof” or “partial progress.”
- Please reference the course collaboration policy [here](#).

For convenience, we restate some definitions used in this problem set.

¹We will assign a minor deduction if we need to maneuver around the wrong PDFs. Please also note that depending on if/how you use Overleaf, you may need to recompile your solutions in between downloads to get the right files.

Problem 1: Linear Programming (20 points, no collaboration)

Alice is trying to get enough oranges and bananas to host a fruit party. To successfully host a party she needs **at least 4 oranges** and **at least 3 bananas**. Unfortunately, her local grocery store only sells fruit in bundles. Bundle A costs 7 dollars and contains 2 oranges and 5 bananas. Bundle B costs 4 dollars and contains 3 oranges and 2 bananas. Fortunately, the grocery store will allow Alice to buy fractions of bundles (i.e. she can buy 2.5 bundle As). They will not allow Alice to buy negative bundles (i.e. she cannot buy -1 bundle As and 3 bundle Bs).

Alice would like to buy x_A bundle As and x_B bundle Bs to guarantee she has at least 4 oranges and at least 3 bananas. Moreover, she would like to find the solution that minimizes her dollars spent.

Part a (10 points)

Write a linear program whose solution is the optimal choice of x_A, x_B for Alice's problem.

Part b (10 points)

Take the dual of the linear program from part a.

Problem 2: Repeated Prisoner's Dilemma (50 points)

Consider the following payoff matrix for prisoner's dilemma. The left number in each entry denotes the payoff for Alice, and the right denotes the payoff for Bob. The left two strategies denote Alice's actions, and the top two strategies denote Bob's.

(Alice, Bob)	Cooperate	Defect
Cooperate	(2,2)	(0,5)
Defect	(5,0)	(1,1)

In this problem, we'll consider the case where Alice and Bob repeatedly play prisoner's dilemma for 1000 rounds. That is, for all t from 1 to 1000, Alice and Bob each choose to Cooperate or Defect. Based on their actions, they each receive some payoff according to the payoff matrix above. When deciding which strategy to use in round t , Alice and Bob each know the entire history of decisions made by the other player (and themselves) for the first $t - 1$ rounds. We'll call this game *Repeated Prisoner's Dilemma*.

Observe that a complete strategy for Repeated Prisoner's Dilemma must decide for all t , as a function of the history from the first $t - 1$ rounds, which action to play in round t .

Part a: No Dominant Strategies (10 points)

Prove that neither Alice nor Bob has a weakly dominant strategy in Repeated Prisoner's Dilemma. See Lecture Game Theory I for a formal definition of weakly dominant strategies.

Hint: Make sure you've understood exactly what a strategy is for Repeated Prisoner's Dilemma, and what it would mean for that strategy to be dominant.

Part b: Iterated Deletion of Dominated Strategies (20 points)

Recall that iterated deletion of weakly dominated strategies repeatedly finds a strategy for some player that is weakly dominated and deletes that strategy (possibly updating which strategies of the other player are weakly dominated), and terminates when both players have no weakly dominated strategies remaining. Recall also that this process is not well-defined, because it may terminate differently depending on how you execute this process (i.e. how you choose which strategies to delete). See Lecture Game Theory I for a formal definition of iterated deletion of weakly dominated strategies.

Prove that there exists an execution of iterated deletion of weakly dominated strategies that terminates with Alice and Bob each only having a single remaining strategy (and state that strategy).

Hint: Again, make sure you've understood exactly what a strategy is for Repeated Prisoner's Dilemma, and shown that *every* strategy (except the remaining one) must be deleted.

Part c: Tit-for-Tat (20 points)

Prove that if Alice plays the "Tit-for-Tat" strategy (defined below), then *no matter what Bob does*, Bob's total payoff after 1000 rounds is at least as large as Alice's payoff after 1000 rounds.

Definition 1 (Tit-for-Tat). *The Tit-for-Tat strategy plays Cooperate in round one. In any round $t > 1$, it copies its opponent's action from round $t - 1$.*

Problem 3: Noisy Cascades (60 points)

Recall the model of information cascades from Lecture Information Cascades: I have a bag containing some number of colored balls. With probability $1/2$ the bag contains 2 red balls and 1 blue ball, and with probability $1/2$ the bag contains 2 blue balls and 1 red ball. One by one, people observe a ball randomly selected from the bag, and guess (based on this observation and all prior guesses) whether the bag has more red balls or blue balls.

In this problem we'll examine a variant of this game. Assume that all players are perfectly rational and guess to maximize their probability of being correct, and know that all other players are perfectly rational and are guessing to maximize their probability of being correct. If the player ever believes that both cases are equally likely, assume that they tie-break to guess the color of their own draw.

The difference between this problem and lecture is that each person's guess is viewed *noisily*; that is, with probability α , each player's true guess is replaced by a uniformly random color before it is announced to the other players (and everyone knows α , and everyone knows that everyone knows α). But each player is aware of their own draw, without noise.

To be extra clear: if player one truly guesses blue, then this is turned into a viewed guess of blue with probability $1 - \alpha/2$ and a viewed guess of red with probability $\alpha/2$. The other players witness this viewed guess (and it is the same viewed guess for all players that view it), and have no other information about the true guess.

Note: Most parts of this problem involve non-trivial calculations. You should both “show your work” for any non-trivial calculations, and also state in text why your calculations are correct (e.g. if you apply Bayes' rule, you should state which equality follows from Bayes' rule. You do not need to argue why your algebra/formula manipulation is correct).

Part a (10 points)

Assume that each of the first k players makes a true guess equal to their draw from the urn. Let $S = \langle G_1, \dots, G_k \rangle$ be an ordered sequence of viewed guesses, where each G_i is either red or blue, and let exactly B of the k revealed guesses be blue, and $R = k - B$ of the revealed guesses be red. What is the probability, as a function of α , *conditioned on the bag containing two red balls and one blue ball*, that you will observe exactly the sequence S of guesses?

Hint: First, try to compute the probability that a particular viewed guess G_i is red, conditioned on the bag containing two red balls and one blue ball. As a sanity check, make sure your formula is correct when $\alpha = 1$ and $\alpha = 0$.

Part b (10 points)

Assume that each of the first k players makes a true guess equal to their draw from the urn. Let $S = \langle G_1, \dots, G_k \rangle$ be an ordered sequence of viewed guesses, where each G_i is either red or blue, and let exactly B of the k revealed guesses be blue, and $R = k - B$ of the revealed guesses be red. Imagine that you have seen exactly the sequence S of viewed guesses before you, and you yourself draw a red ball from the urn. Conditioned on this, what is the probability, as a function of α , that the bag contains two red balls and one blue ball?

Hint: Use Bayes' rule. Part a helps you find the probability of seeing S (plus you draw a red) conditioned on there being two red balls and one blue ball.

Part c (15 points)

Assume that each of the first k players makes a true guess equal to their draw from the urn. As a function of α , how many more red viewed guesses than blue viewed guesses (or vice versa) would you have to see before you decide to ignore your own draw?

Use this to briefly argue that, for all $\alpha < 1$, a cascade will *eventually* happen. Also briefly argue that when $\alpha = 1$, a cascade will never happen. This brief argument can be just a few sentences, and is not expected to have any mathematical formality.

Hint: The math will be simpler if you try to argue that you believe it is more likely the urn is red versus blue, rather than trying to argue that the probability that the urn is red is $> 1/2$.

Part d (10 points)

When $\alpha < 1$, let $S = \langle G_1, \dots, G_k \rangle$ be an ordered sequence of viewed guesses so that a red cascade begins right after viewed guess k (that is, the $(k + 1)^{st}$ player will ignore their draw). Let S' be the ordered sequence S , but with all reds and blues flipped (that is, all reds are now blue, and blues are now reds).

As a function of α , what is $\Pr[S|\text{two reds one blue}] / \Pr[S'|\text{two reds one blue}]$?

Part e (15 points)

When $\alpha < 1$, as a function of α , what is the probability that the eventual cascade is correct?

Extra Credit: Another Algorithm for Nash

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.²

For this problem, you *may* collaborate with any students. You *may not* consult course resources or external resources. In this problem we will guide you through the proof of a well-known result, so you should *not* copy the proof from one of the course texts (nor should you try to find a proof from external sources). You *must* follow the guide below (and not provide an alternative proof).

Consider any *symmetric* two-player game. That is, $p_1(x, y) = p_2(y, x)$ for all x, y . Consider also the following system of inequalities. We'll refer to this as the *Lemke-Howson Polytope*.

- Variables x_1, \dots, x_n .
- (Non-negativity) $x_i \geq 0$ for all i .
- (Responsiveness) $\sum_j x_j p_1(i, j) \leq 1$.

Part a

Say that an action i is *covered in* \vec{x} if either the non-negativity constraint is tight (i.e. $x_i = 0$) or the Responsiveness constraint is tight (i.e. $\sum_j x_j p_1(i, j) = 1$), or both. Prove that if \vec{x} is inside the Lemke-Howson polytope, and $\vec{x} \neq \vec{0}$, and all actions i are covered in \vec{x} , then $\vec{x}/|\vec{x}|_1$ is a symmetric Nash equilibrium (that is, $\vec{x}/|\vec{x}|_1$ is a best response to itself).

Part b

For this part, you should assume that any set of n equations taken above have a solution, and that this solution is unique.

The Lemke-Howson algorithm starts from the point $\vec{0}$ and repeatedly *pivots*. That is, the current point will always have exactly n tight constraints. The pivot will pick one of these constraints and “relax” it (keeping the other $n - 1$ tight). A new constraint will become tight, and this will be the new point (you do not need to prove that this procedure is well-defined).

From $\vec{0}$, the pivot rule simply picks an arbitrary tight constraint to relax (let's say $x_1 = 0$). This causes a new constraint to become tight. If it's the 1st Responsiveness constraint, then by Part a we've found a Nash and are done! If not, then we have exactly one *double-covered* action. That is, there is some action i such that the non-negativity and Responsiveness constraints are both tight. We pick the non-negativity constraint for i to relax next.

In general, for our current point $\vec{x} \neq \vec{0}$, if there is no double-covered action we terminate (and hope that it's a Nash and not back at $\vec{0}$). If there's a double-covered action, it's because we just made one of the constraints for i tight. So relax the other one and continue.

Prove that the Lemke-Howson algorithm will never revisit a vertex \vec{y} without first revisiting the origin. You may use without proof the fact that if \vec{z} pivots to \vec{w} when constraint C is relaxed,

²Keep 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.

causing constraint D to become tight, then \vec{w} pivots to \vec{z} when constraint D is relaxed, causing constraint C to become tight.

Part c

Prove that the Lemke-Howson algorithm cannot ever return to $\vec{0}$. Conclude that the Lemke-Howson algorithm finds a Nash after at most $\binom{2n}{n}$ pivots.

Hint: You may want to prove that the algorithm terminates as soon as 1 becomes covered.