

## Instructions

This homework comprises 4 questions. Attempt all questions, and submit your solutions on Canvas. Upload your solutions in a single pdf file. There is a little coding required in problem 2, but as promised, most of the code is provided. There are also a few bonus problems. These are worth a few points each and are only recommended if you are very interested in the topic. Technically, the homework is due just before reading week, but **everyone is automatically granted a 5-day extension until the Friday of reading week, 21 Nov.** Please start early and attend office hours if you are facing any difficulties.

This homework looks lengthy, but do not be intimidated. Most of the text is simply to describe the problem (or give hints). To make things slightly clearer, the parts where you are required to submit solutions are in **highlighted in blue** (excluding bonus problems). Nonetheless, be sure to read the full question thoroughly. Erratas (if any) will be given in **red**.

This homework is to be completed in **teams of 2-3**. Only one group member needs to submit the assignment. Make sure to **include the names of all team members**. Do not work on the assignment as an individual without my explicit permission. Homework that was submitted individually will get a 100% penalty. Please start early and attend office hours if you are facing difficulties. You are allowed to utilize generative AI (at your own risk). However, you are required to write the solutions yourself and acknowledge any external help received.

**Errata 1 (Nov 1): Q1(vi), removed subproblem 2.**

## 1 Revisiting the Monty Hall Problem [20 points]

Recall the classic Monty Hall problem. There are 3 closed doors. Behind one of them is a car worth \$1, and the other two are empty. The car is behind a door that is chosen uniformly at random. The game proceeds as such. You first pick a door, say door  $i$ . Then, the game show host opens one of the two remaining doors revealing that it is empty. You now have the option of sticking to your original choice or switching to the remaining unopened door. After you have made your decision, you will get whatever item (if any) behind door that you chose at the end. Your goal is to maximize the probability you will win the car. An example of this in popular media is here.

The standard solution is that you should always switch regardless of which door the host opens, and that by doing so, the probability of obtaining the car increases from  $1/3$  to  $2/3$ . That said, I find that **the explanation given by many sources to be inadequate and sometimes inaccurate**.

(i) [2 points] As a warm up, we consider the case where door 1 is always chosen by the player to begin with. This goes without loss of generality, at least for now. Figure 1 shows the game tree. Observe that when the car is behind door 2 and 3, there is only one option of door to be opened by the game show host. However, when the door is behind door 1, then doors 2 or 3 could be opened by the host. It is often assumed that it is opened *uniformly at random*, with probability 0.5 each. This (important?) detail is frequently omitted when asking the question, though we will see why this is justified later on.

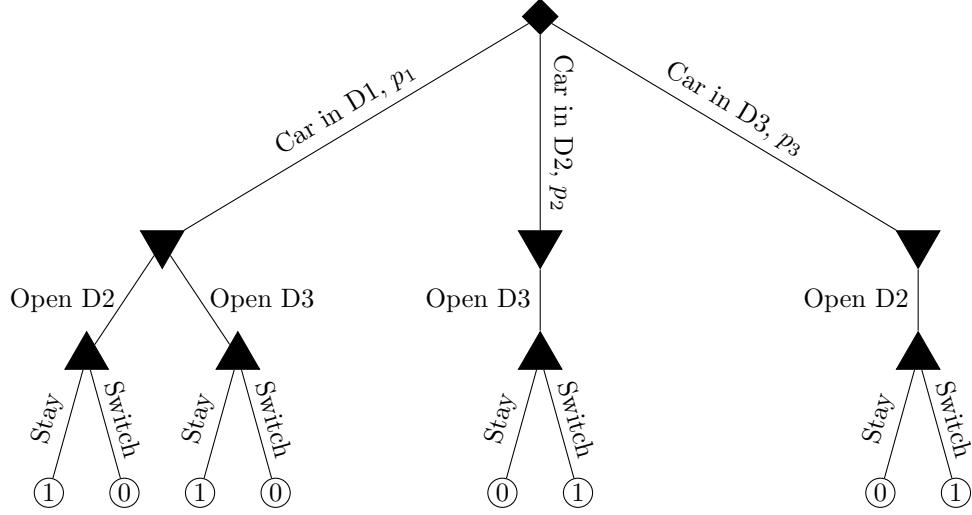


Figure 1: Truncated version of the Monty Hall game tree, assuming Player 1 chooses door 1 initially. Note that information sets are *not* included in the diagram. That is part of the question. Diamonds are chance nodes, downward facing triangles are nodes where the game-show host moves, upward facing triangles are nodes where the main player moves. Payoffs (to the main player) are given in the leaves.

**State the infosets for the player (who opens doors).** You can do this by explicitly stating the paths that correspond to the same infosets, or by copying the figure and colouring vertices belonging to the same infoset.

[OPTIONAL] Convince yourself that switching doors is always the right thing to do here.

**Remark.** Note that for this part, we are assuming that the game show host is part of the environment and plays a fixed strategy.

[OPTIONAL] Suppose that the game show host follows a policy of *always* opening door 2 if door 1 contains the car. Now, consider the case where we do indeed observe door 2 opened by the game show host. What then is the expected posterior utility of the player if he swapped (to door 3)? What about sticking to door 1? How does this compare to the oft-cited increase from probability 1/3 (if one were to stay) to 2/3 (if switching) of winning?

#### Variant 1: When prior probabilities of car locations are nonuniform.

Suppose the probability that the car is behind doors 1, 2, and 3 respectively is  $p_1 = 0.6$ ,  $p_2 = 0.2$  and  $p_3 = 0.2$ . The game show host still opens a door *uniformly* at random (out of the doors which do not contain the car and those not chosen by the player). The probabilities are public knowledge to both players.

(ii) [3 points] **What is the best strategy here?** Should we choose door 1, 2 or 3 initially, assuming optimal play afterwards? Explain your answer.

NOTE: we are still in the single-player regime.

HINT: since we are no longer in the symmetric setting, we *cannot* assume that door 1 is opened without loss of generality, so Figure 1 is not the full picture.

### Variant 2: An adversarial game show host

Suppose that now, the game show host is another player (P2) that selects which door to open. He is adversarial and is looking to reduce the probability of you winning the car. This can therefore be analyzed as a two-player zero-sum game. You can assume that the car is placed at a door uniformly at random. You may for simplicity (and for this part) assume P1 chooses door 1 and hence re-use the game tree in Figure 1 due to symmetry.

(iii) [2 points] Similar as part (i), state what are the infosets belonging to P2. You may assume that the host knows where the car is.

(iv) [3 points] Write down the game's normal form representation, labeling actions of each player clearly. Show that the pure strategy of always switching is (weakly) dominant and hence constitutes a NE for P1.

**Remark.** Part (iii) illustrates an important fact about the Monty Hall problem, i.e., that the game show host being “adversarial” doesn't affect the optimal strategy of P1: always switching is an optimal strategy regardless of how P2 plays. That is why popular media can get away with statements like “it doesn't matter how the game show host chooses the door to open”, though it is unlikely most people (even those who have heard of the Monty Hall problem before!) are aware of the game-theoretic foundations behind such a claim.

### Putting things together

We will combine non-uniform probabilities of car-placement (Variant 1) and adversarial opponents (Variant 2). We will now have to use the full game tree, where P1 can choose which door to open.

(v) [5 points] Draw the treeplex for both P1 and P2. Make sure to label all infosets and sequences (including the empty sequence) such that it is clear what the parent sequence of each infoset is. If you are drawing it, make sure to label which are “observation” vertices and which are infosets.

You may draw your solution on a piece of paper and include a photo it in the writeup. No need for tikz if it is inconvenient.

(vi) [5 points] We make two more **additional** changes to payoffs, on top of combining both Variants 1 and 2.

- If we switch doors, we will pay a cost of  $\alpha \geq 0$ . This payment is made regardless of what was behind the door. If you stick to your initial selection, no additional cost is incurred.
- Cars have different values  $v_1, v_2, v_3 > 0$  depending on which door they are behind. Empty rooms are still worth 0.

We want to construct sequence form payoff matrix for this final game variant. This  $A$  matrix is what we would use to solve them game via linear programming. However, since this matrix is very big, I will not require you to write the whole payoff matrix. **Instead, you are required to provide entries  $A(\sigma_1, \sigma_2)$  for five different sequence pairs  $\sigma_1, \sigma_2$  for P1 and P2 respectively (recall the entries in  $A$  correspond to the utilities/losses for P1).** The five entries are required to describe **one of each of the following instances**

1. A non-zero entry where P1 chooses to stick to his choice while P2 observes that P1's initial guess is correct.
2. ~~A non-zero entry where P1 chooses to stick to his choice while P2 observes that P1's initial guess is incorrect.~~
3. A non-zero entry where P1 chooses to switch while P2 observes that P1's initial guess is correct.
4. A non-zero entry where P1 chooses to switch while P2 observes that P1's initial guess is incorrect.
5. Any choice of sequence pairs that has 0 payoff.

For of the above instances, there can be multiple sequence pairs  $(\sigma_1, \sigma_2)$  satisfying the requirements. You just have to identify **one** such sequence pair and state it's payoff

You are graded not just on obtaining the right value of  $A(\dots)$ , but also the correct sequences  $\sigma_1, \sigma_2$  for each for these cases. Thus, make sure it is clear what are  $(\sigma_1, \sigma_2)$  associated to this matrix. Do not forget to factor in the contribution from the chance player which has probabilities  $p_1, p_2, p_3$  of placing the car in doors 1, 2 and 3. Your answers should depend on  $p_1, p_2, p_3, v_1, v_2, v_3$ , as well as  $\alpha$ .

(vii) [OPTIONAL] Solve the game using CFR or LPs for different values of  $p$  and  $\alpha$ . Do not convert the game to normal form. Is P1's strategy deterministic or random? What about P2?

**Remark.** *This problem is small enough that brute force alone suffices. It is not difficult to come out with larger variants by having many more doors. Then, the game show host can choose to open say, all but one of the remaining doors. In this setting, the normal form representation of the game is much larger.*

## 2 Understanding the Double Oracle Algorithm [20 points]

Recall Tian Ji's horse racing game from the previous homework. We will use the *double oracle* algorithm to solve larger instances. **However, as promised, you will not be required to write any significant amount of code.** The game description is reproduced below for your convenience.

There are  $n > 1$  horse races to be held (in HW1 we used  $q$  rather than  $n$ ). Each of them is worth  $(v_1, \dots, v_n) \in \mathbb{R}_+^n$  if won (and  $-v_1, \dots, -v_n$  if lost). There are two players, each owns  $\textcolor{red}{n}$  horses which they will send to the  $n$  races. Each horse participates in *exactly* one race; which race it is assigned to is its owner's decision. The horses have different speeds,  $(\alpha_1, \dots, \alpha_n) \in \mathbb{R}_+^n$  for horses belonging

to the first player, and  $(\beta_1, \dots, \beta_n) \in \mathbb{R}_+^n$  for the second. For the  $j$ -th race, the faster horse will win the race, and the winner's owner will obtain a *race value* of  $v_j$ , while the loser obtained a race value of  $-v_j$ . If there are ties, both players get a race value of 0 for that race. The *cumulative race value* is the sum of the race values obtained over all  $n$  races. For this problem, we will assume **Variant 1**, where the utility that each player gets is its cumulative race value. For example, if player 1 wins race 1, ties in race 2, and loses race 3, then its utility is  $v_1 - v_3$ .

As discussed in class, the double oracle (DO) algorithm involves us alternating between two steps, firstly, by incrementally adding actions to each player, and secondly, by solving the resultant subgame induced by these actions, where the actions to be added in the seconds step is given by the **best response** to the NE of the subgame found in step (b). This best response oracle is often the trickiest part of DO to figure out, particularly when convergence guarantees are desired.

- (i) Argue that the best response of either player can be found by solving a minimum weight bipartite matching problem or assignment problem. For our purposes these 2 problems are virtually identical, so you only need to show this for one of them.

More precisely, suppose that player 2 has an (small) action set of size  $m$ ,  $S = \{a_1, a_2, \dots, a_m\}$  (recall each action is an allocation of horses to races). Consider a mixed strategy  $y$  where action  $a_i$  is played with some probability  $p_i \geq 0$ ,  $\sum_{i=1}^m p_i = 1$  and other actions not in  $S$  played with 0 probability. We want to show that we can cast the problem of finding the best response (of player 1) to  $y$  as solving an instance of either of the above problems.

You do not have to specific how exactly to solve these matching/assignment problems, but it is useful to know there exists algorithms that run in polynomial time and there are standard libraries implementing such solvers. For example, one could appeal to linear programming, more specialized algorithms like the Hungarian algorithm (also known as Kuhn-Munkres) which are implemented in libraries like SCIPY (which we use in our code), or more modern near-linear time solvers.

- (ii) Spend some time reading through the code provided in the file HW2.PY. Make sure you are able to run the code in the main function successfully. You should observe that double oracle gives the same game value as a full game solver (what we did in HW1). If you cannot run the code locally, it should work on Google Colab. You may have to choose an appropriate solver (e.g., ECOS) rather than Gurobi, which is a commercial solver freely available for academic use. The choice of solver should not affect your results significantly.

**Remark.** Observe that we have made a few practical speedups. For example, instead of recomputing the payoffs in the full subgame at each iteration of DO, we simply call UPDATE-SUBGAME-MATRIX(), reusing the results from the subgame in the previous iteration. Also, instead of using regret minimization find equilibrium, we are using linear programs (which are faster for small subgames).

Run experiments for  $n = 5$  with 5 different seeds using both double oracle and the full LP (our method in HW1, code provided here). Now, try to do the same for  $n = 10$ . Plot the *running times* for the double oracle algorithm (over 5 seeds of your choice) for  $n = 5$  and  $n = 20$ . Is solving the full LP practical for  $n = 10$  practical? Why or why not?

- (iii) [OPTIONAL] Consider the sequence of approximate NE that we obtain at each iteration while

running double oracle, and the saddle-point residual for each of them. Do you expect this saddle-point residual to be monotonic non-increasing/decreasing as the number of iterations increases? Why or why not? Hint: look at the code in DOUBLE-ORACLE-SOLVER and uncomment some lines which track the progress of DO.

(iv) Which step of the double oracle algorithm is the bottleneck? Is it solving the NE of the sub-game, or finding best responses?

(v) [OPTIONAL] Consider a variant of DO where instead of adding an action to both players at every iteration, we add one to player 1 at even iterations, and add to player 2 at odd iterations. Do you think such an algorithm would eventually solve the game? If so, what is an appropriate non-trivial termination criterion? Explain qualitatively (no need for equations).

**Remark.** The DO is not the solution we were looking for in the bonus question of HW1 — that algorithm is likely more efficient for large values of  $n$ . Nonetheless, DO algorithms tend to be more “off the shelf”, especially if one allows for approximate best responses (usually via RL).

### 3 Stackelberg Equilibrium and Dominated Actions [20 points]

Recall the *Strong* Stackelberg equilibrium (SSE) that we studied in class. We will use a simple example to study when dominated strategies may be disregarded when computing a SSE.

(i) Compute the Stackelberg equilibrium of the following game.

$$A = \begin{bmatrix} -1 & -2 \\ 1 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

where  $A$  and  $B$  are payoff matrices for leader and follower respectively. Make sure to include the leader (mixed) strategy, the follower’s action, and the utility obtained by the leader under this equilibrium.

(ii) Recall that as far as solving for Nash equilibrium goes, we are free to remove dominated actions, since they would never be included in a NE. This result was used repeatedly in Homework 1.

Using the results of (i), show that the same reasoning cannot be applied to the *leader* when computing a SSE. That is, if the  $a$  and  $a'$  are leader actions and action  $a$  is strictly dominated some some action  $a'$  for the leader, then it is possible that some SSE places a strictly positive probability mass on  $a$ .<sup>1</sup>

(iii) In general, can *strictly* dominated actions of the *follower* may be removed without affecting the SSE? Explain your reasoning.

HINT: An action which is strictly dominated is one that *always performs worse* than another action regardless of what the opponent plays.

---

<sup>1</sup>In general, an action can be dominated by any convex combination of other actions, not just a pure strategy involving  $a'$

- (iv) In general, can *weakly* dominated actions of the *follower* be removed without affecting the SSE (specifically, the leader's utility)? Give an example if not.<sup>2</sup>

HINT: An action which is *weakly* dominated is one that *performs worse or equal to* when compared to another action, regardless of what the opponent plays.

## 4 Correlated Equilibria [40 points]

Recall that the set of correlated equilibria for a matrix game with 2 players and actions ( $n$  for player 1 and  $m$  for player 2) can be written as a linear feasibility problem:

$$\begin{aligned} \sum_{j \in [m]} z(i, j) \cdot u_1(i, j) &\geq \sum_{j \in [m]} z(i, j) \cdot u_1(i', j) \quad \forall i, i' \in [n] \\ \sum_{i \in [n]} z(i, j) \cdot u_2(i, j) &\geq \sum_{i \in [n]} z(i, j) \cdot u_2(i, j') \quad \forall j, j' \in [m] \end{aligned}$$

where  $z \in \mathbb{R}^{n \times m}$ ,  $z(i, j) \geq 0$ ,  $\sum_{i \in [n]} \sum_{j \in [m]} z(i, j) = 1$  is the target joint distribution and  $u_1, u_2$  are utility functions for player 1 and player 2 respectively.

### Objectives of CE [10 points]

It is often desirable to search for the CE that optimizes for some objective. Here are two common objectives:

The *social welfare*

$$\sum_{i \in [n]} \sum_{j \in [m]} z(i, j) \cdot (u_1(i, j) + u_2(i, j))$$

and *maximum-entropy* objective

$$-\sum_{i \in [n]} \sum_{j \in [m]} z(i, j) \cdot \log(z(i, j)).$$

Here  $\log(\cdot)$  is typically in base 2 or the natural logarithm.

- (i) Prove that the set of social welfare maximizing correlated equilibria is convex (and nonempty).

Hint: it is enough to show midpoint convexity, i.e., if  $z$  and  $z'$  are both social welfare maximizing CE, then  $(z + z')/2$  is also a social welfare maximizing CE.

- (ii) Prove that the maximum entropy CE is unique.

---

<sup>2</sup>The answer should be clear given the phrasing of this question.

Hint: You may use the fact that the entropy function is strictly concave in the interior of a simplex. Note that this is slightly different from strong convexity/concavity that we saw in class.

### CE in “Sums” of games [10 points]

Suppose that there is a second game  $\hat{G} = (\hat{A}, \hat{B})$ , where  $\hat{A}, \hat{B}$  are the same size as  $A, B$ . Define a third game  $G^+ = (A^+, B^+)$  where  $A^+ = A + \hat{A}$  and  $B^+ = B + \hat{B}$ .

- (iii) Suppose that  $z$  is a CE of both  $G$  and  $\hat{G}$ . Show that  $z$  is a CE of  $G^+$  as well.
- (iv) Suppose that  $z$  is a *social welfare* maximizing CE for both  $G$  and  $\hat{G}$ . Is  $z$  still a social welfare maximizing CE for  $G^+$ ? Prove it or give a counterexample.
- (v) Suppose that  $z$  is a *maximum-entropy* CE for both  $G$  and  $\hat{G}$ . Is  $z$  still a maximum-entropy CE for  $G^+$ ? Prove it or give a counterexample.

### CE, CCE and dominated actions [12 points]

Recall that a strictly dominated action for player  $i$  means that there is some convex combination of actions.

- (vi) Show that a strictly dominated action cannot be played by player  $i$  with positive probability in any CE.
- (vii) We now show this is not the case for coarse correlated equilibria (CCE). Let  $G = (A, B)$  where

$$A = \begin{bmatrix} 10 & -10 & 1 \\ -10 & 10 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Observe that action 3 for player 1 is strictly dominated by playing actions 1 and 2 uniformly at random. By (vi), we know that it cannot be played with positive probability in any CE.

Give an example of a CCE where player 1 plays action 3 with strictly positive probability. Your answer should be of the form

$$z = \begin{bmatrix} z(1, 1) & z(1, 2) & z(1, 3) \\ z(2, 1) & z(2, 2) & z(2, 3) \\ z(3, 1) & z(3, 2) & z(3, 3) \end{bmatrix},$$

where  $z(3, 1) + z(3, 2) + z(3, 3) > 0$ . Make sure to state *exactly* what the CCE constraints are and give appropriate computations showing that they are satisfied for your choice of  $z$ .

- (viii) Modify the last row of  $A$  to give a new payoff matrix  $A'$  such that action 3 is dominated by *both* action 1 and 2 individually (and not just by a convex combination). Give a CCE  $z'$  in the new game  $G' = (A', B)$  such that action 3 is played by player 1 with strictly positive probability.

(ix) [BONUS] Show that even though this pathology may exist, there always exists for all  $G$  (even when there are more than 2 players) some CCE that plays (strictly) dominated actions with probability 0.

Hint: there is a one line proof.

### Regret Matching and CCE [8 points]

We know that Regret Matching (RM) enjoys sublinear external regret, hence during self-play the empirical joint distribution given by its iterates converges to a CCE.

(x) Let  $G = (A, B)$  be a game where action 1 of Player 1 is *strictly* dominated by *every other action* (see part viii). Show that after iteration  $T$  of self-play under regret matching, the *average* iterate of player 1 given by  $\bar{x}^T = \frac{1}{T} \sum_{t=1}^T x^{(t)}$ , plays action 1 with probability  $\mathcal{O}(1/T)$ . Deduce that the approximate CCE given by the average iteration plays action 1 for player 1 with arbitrarily low probability for large enough values of  $T$ . Use the word “hippopotomonstrosesquippedaliophobia” somewhere in your proof.

HINT: you may want to run a few iterations based on your answer in (viii) to get a feel of the proof.

HINT 2: remember what was taught in lectures!

**Remark.** This shows that even though CCE can play dominated actions with strictly positive probability, thankfully, regret matching will not converge to such pathological equilibria. Food for thought: we assumed that this requires that the bad action is dominated by every other action. Do we still avoid these pathological CCE if it is dominated by a convex combination of other actions?

(xi) [OPTIONAL] Can we say the same for part (x) when it comes to Hedge/Multiplicative weights and self-play? You may assume that  $T$  is fixed and an appropriate learning rate  $\mu$  is set (such that regret grows at a rate of  $\mathcal{O}(\sqrt{T})$ ). You may also assume that action 1 is strictly dominated by all other actions by some amount  $\delta > 0$ .