Computational Microeconomics

Homework 5: Bayesian games and Generalized Vickrey Auction (due Dec. 6 before 5pm)

Please read the rules for assignments on the course web page (`http://www.cs.cmu.edu/~15326-f24/`). Use Piazza for questions and Gradescope to turn this in. For all questions, always hand in both code and output, typically .mod and .out files (and do not simply put everything in a .pdf).

Please use clear variable names and write comments in your code where appropriate (you can put comments between `/*` and `*/`, or start a line with `#`).

1. Funding a project.

There is a project that needs funding. Two agents can fund the project. Each agent has two choices: 1. Pay 1 to fund the project; 2. Pay 0. Depending on funding, the project may succeed or fail. If the total amount of money paid by the agents is 2, then the project will succeed with probability 1. If the total amount is 1, then the project will succeed with probability 1/3. If the total amount is 0, then the project will succeed with probability 0.

Each agent i ($i \in \{1, 2\}$) has a valuation v_i for a successful project. (Failed projects are worthless.) This valuation does not depend on whether the agent contributed or not (the agent can make use of the project regardless). Each valuation is drawn uniformly at random from $[0, 4]$.

For example, if agent 1 draws valuation 2 and pays 0, and agent 2 draws valuation 3 and pays 1, then:

- The probability of a successful project is 1/3;
- Agent 1's expected utility is $(1/3) \cdot 2 - 0 = 2/3$;
- Agent 2's expected utility is $(1/3) \cdot 3 - 1 = 0$.

(a) Show that the following strategy is a Bayes-Nash equilibrium of this game (if both players use it): pay 1 if and only if your valuation is at least 2. (Hint: show that if the player's valuation is 2, then she is indifferent between the two actions, given that the other player uses this strategy.)

(b) More generally, suppose that each player's valuation is drawn (independently) from some arbitrary distribution with cumulative density function F (but keep everything else the same). Show that the following is a Bayes-Nash equilibrium of this game (if both players use it): pay if and only if your valuation is at least x, where x is the solution to $F(x) \cdot x - 2x + 3 = 0$.

2. Generalized Vickrey Auction (= Clarke mechanism on combinatorial auctions).

Consider the following 3 bids in a combinatorial auction with 3 items (with free disposal):

$(\{a, b\}, 10)$ XOR $(\{c\}, 4)$
$(\{a, b\}, 6)$ XOR $(\{b, c\}, 9)$
$(\{a\}, 3)$ XOR $(\{a, b, c\}, 11)$

Solve the winner determination problem, and compute the GVA (Clarke) payments for the winning bidders. (Remember to remove each bidder's *entire* bid when calculating her payment.)

3. A compact LP for two-player zero-sum Bayesian games.

For most of the Bayesian games that we studied in class, a player only cares about her own type, and not about the types of the other players (except for how those types influence the behavior of the other players). But in general, a player can care about other players' types as well. In a two-player zero-sum Bayesian game, if one player cares about her type, then the other player *must* care about that type as well, since the game is zero-sum. But each player only *sees* her own type. For example, you might imagine a card game in which both players draw cards once at the beginning (your type is your hand of cards), then simultaneously each take an action, and then the cards and the actions together determine the payoffs.

How can we solve for a maximin strategy for player 1? We could convert the game to normal form and then use our usual linear program for maximin strategies. However, the normal form in general is going to have exponential size, so that is not the best approach. Instead, you will develop a more compact linear program in this problem.

You should have parameters $u(\theta_1, \theta_2, a_1, a_2)$ (this is the utility of player 1; we don't need to have a separate parameter for player 2's utility because the game is zero-sum) and $P(\theta_1, \theta_2)$ (the *joint* probability of θ_1, θ_2; we will allow them to be correlated). You should have a variable $\sigma(\theta_1, a_1)$ for the probability player 1 places on a_1 if she has type θ_1. You should also have a variable v_{θ_2} for the value player 1 will get from the case where player 2 has type θ_2 (i.e., how much this case contributes to the total expected value of the whole game); the sum of these should be the overall value that player 1 gets out of the entire game. You should have a constraint for every combination of θ_2 and a_2 to make sure the v_{θ_2} variables take sensible values, given that player 2 tries to minimize these. Finally, you should have a constraint for every θ_1 that the σ probabilities add up to 1 for that type.

(Notice that the types of things that we have variables for for the one player are the types of things that we have constraints for for the other player. This is no accident; in the dual of the linear program, the roles will be reversed.)

Hint: If player 2 uses (pure) strategy $s(\theta_2)$, then player 1's utility can be expressed as $\sum_{\theta_2} \left[\sum_{\theta_1} P(\theta_1, \theta_2) \sum_{a_1} \sigma(\theta_1, a_1) u(\theta_1, \theta_2, a_1, s(\theta_2)) \right]$. The part between the right brackets ([and]) should correspond to v_{θ_2} if s is a best-response strategy to σ.

You should complete the following linear program and test it with the instance in it (Instance 1). But do NOT turn it in with this instance; you should turn it in with Instance 2, further below.

```
set THETA1;  # Types of player 1
set THETA2;  # Types of player 2

set A1;      # Actions of player 1
set A2;      # Actions of player 2

param u{THETA1, THETA2, A1, A2};  # Utility of player 1
param P{THETA1, THETA2};          # Joint probability of types

var sigma{THETA1, A1} >= 0;  # Strategy of player 1
var v{THETA2};               # Value for each type of player 2

# Objective: Maximize player 1's total expected value
maximize total_value:        # YOUR TASK IS TO COMPLETE THIS

# Constraints
s.t.        # YOUR TASK IS TO COMPLETE THIS

data;

set THETA1 := High Low;  # Types of player 1 (High card or Low card)
set THETA2 := High Low;  # Types of player 2 (High card or Low card)

set A1 := Bet Fold;      # Actions of player 1
set A2 := Bet Fold;      # Actions of player 2

param u :=
[High,High,*,*]:  Bet    Fold :=
         Bet      5     10
         Fold    -1     -1

[High,Low,*,*]:   Bet    Fold :=
         Bet      15     5
         Fold    -1     -1

[Low,High,*,*]:   Bet    Fold :=
         Bet     -5      2
         Fold    -1     -1

[Low,Low,*,*]:    Bet    Fold :=
         Bet     -1     -1
         Fold    -1     -1
;

param P:
        High    Low :=
High    0.25    0.25
Low     0.25    0.25;

end;
```

For the above instance (Instance 1) you should get an optimal total value of

2, which is reached with: always Bet when High and always Fold when Low, i.e., sigma[High,Bet]=1, sigma[High,Fold]=0, sigma[Low,Bet]=0, sigma[Low,Fold]=1, v[High]=v[Low]=1.

Now modify the data part as follows, corresponding to Instance 2. Turn it in with this.

```
data;

set THETA1 := High Low;  # Types of player 1 (High card or Low card)
set THETA2 := High Low;  # Types of player 2 (High card or Low card)

set A1 := Bet Fold;      # Actions of player 1
set A2 := Bet Fold;      # Actions of player 2

param u :=

[High,High,*,*]:  Bet    Fold :=
         Bet      8      0
         Fold     0     -2

[High,Low,*,*]:   Bet    Fold :=
         Bet     12      3
         Fold     0     -1

[Low,High,*,*]:   Bet    Fold :=
         Bet     -30      1
         Fold     0     -3

[Low,Low,*,*]:    Bet    Fold :=
         Bet      4     -2
         Fold     0     -1
;

param P:
        High    Low :=
High    0.3     0.2
Low     0.2     0.3;

end;
```