Assignment 4: Strategic Bidding and Revenue in Ad-auctions

Pietro Bonazzi

Economics and Computation, MSc Course, 1st Sem., Fall 2020
University of Zürich
pietro.bonazzi@uzh.ch

Contents

Chapter 1

Designing a bidding agent

Compute a vector of expected utilities for each slot and the optimal bid.

1.1 Expected utilities

The vector of expected utilities include all utilities of the agent i occupying the slot j during each round in the simulation. The formula described in the problem statement has been coded as follows :

```
        for(int i=0; i < numclicks.size(); ++i){
            int utility = numclicks.get(i) *(value - bids.get(i));
            expectedUtils.add(i,utility);}
```

The number of clicks of each slot j and the value of the agent are computed to calculate the utility,

```
        ArrayList<Integer> numclicks = history.getSlotClicks(t-1);
```

```
        int value = getValue();
```

The expected payment for each slot is the average between the biggest and the lowest bid to win each slot which were made in the previous round. Although the agent might win the slot bidding the minimal value, the choice has been to increase the chances of the agent to win the slot and therefore to expected a payment in the middle between the two extremes.

```
        ArrayList<Integer> bids = new ArrayList<>();
        ArrayList<Pair> bidtowin = slotInfo(t, history, reserve);

        for(int i=0; i < numclicks.size(); ++i){
            int bet = (bidtowin.get(i).getFirst()+bidtowin.get(i).getSecond())/2;
            bids.add(i,bet);}
```

1.2 Optimal Bid

The optimal bid made by a balanced bidding agent respects the following : equation

$$b_i = w_i - (q_i^* * (w_i - p_i^*)/q_{i-1}^*)$$

All the variables in the equations have been treated separately and then employed in the final optimisation equation.

```java
        /* qi and qi-1 */
        /* BEGIN */
        ArrayList<Integer> numclicks = history.getSlotClicks(t-1);
        int qi__star;
        int qi_1__star;

        if(targetSlot==0){
            qi__star = numclicks.get(targetSlot);
            qi_1__star = qi__star * 2;}
        else {
            qi__star = numclicks.get(targetSlot);
            qi_1__star = numclicks.get(targetSlot-1);}
        /* END*/

        /* wi */
        /* BEGIN */
        int wi = getValue();
        /* END*/

        /* pi__star */
        /* BEGIN */
        ArrayList<Integer> perClickPaymentsHist = history.getPerClickPayments(t-1);
        int pi__star;
        if (perClickPaymentsHist.size()==0) {pi__star = reserve;}
        else {pi__star = perClickPaymentsHist.get(targetSlot);}
        /* END*/

        /* final optimisation equation*/
        /* BEGIN */
        int testbid = wi - ((qi__star * (wi - pi__star)) / qi_1__star);
        if (testbid>wi) { bid = wi;}
        else {bid = testbid;}
        /* END*/
```

Chapter 2

Agent analysis

Run the simulation with 5 agents and the default budget of 5000.

2.1 Population average and comparison

The table 2.1 below summarizes the findings of two simulations of ad-auctions. Two populations are studied. The first population is constituted by five truthful agents, whereas the second one is made of five balanced bidding agents. In the two simulations the number of iterations was altered in order to detect possible changes in the average utility of individuals in both populations.

Table 2.1: Individuals average utility

maxperms=1 seed=2 numiters=50	
Population	Avg. Utility
TruthfulAgent	24717
BBAgentPIBO	53311,8
maxperms=1 seed=2 numiters=100	
Population	Avg. Utility
TruthfulAgent	23667,4
BBAgentPIBO	53634,8

The BBAgentPIBO's population is averaging a greater utility among its individuals, compared to the population of Truthful agents. An increase in the number of iterations improves the average utility of individuals in the BBAgentPIBO population but not in the TruthfulAgent population.

2.2 Truthful agent performance

In the first scenario, agent 0, a Truthful agent, finds itself in a population of Truthful agents. In table 2.2 its performance is summarized.

Table 2.2: Performance in a T Agents population

maxperms=1 seed=20 numiters=200			
Simulations	TruthfulAgent(0)	Average	Difference
1	22164	23928	-1764

In the second analysis, Agent 0 is placed in a group of BB agents. Table 3.2 illustrates what happened.

Table 2.3: Performance in a mixed population

maxperms=1 seed=20 numiters=200			
Simulations	TruthfulAgent(0)	Average	Difference
1	49172	50232,2	-1060,2

In the second population the averaged utility of Agent 0 has more than doubled. To conclude, a Truthful Agent has a great advantage being with a population of BB Agents.

Chapter 3

Mechanism Analysis

Run the simulation with 5 agents and the default budget of 5000.

3.1 Auctioneer's revenue under GSP as a function of r

The reserve price maximize the revenue of the auctioneer at around 65. After around 85/90 the impact on the auctioneer's revenue is negative. The figure 3.1.1 summarizes the findings.

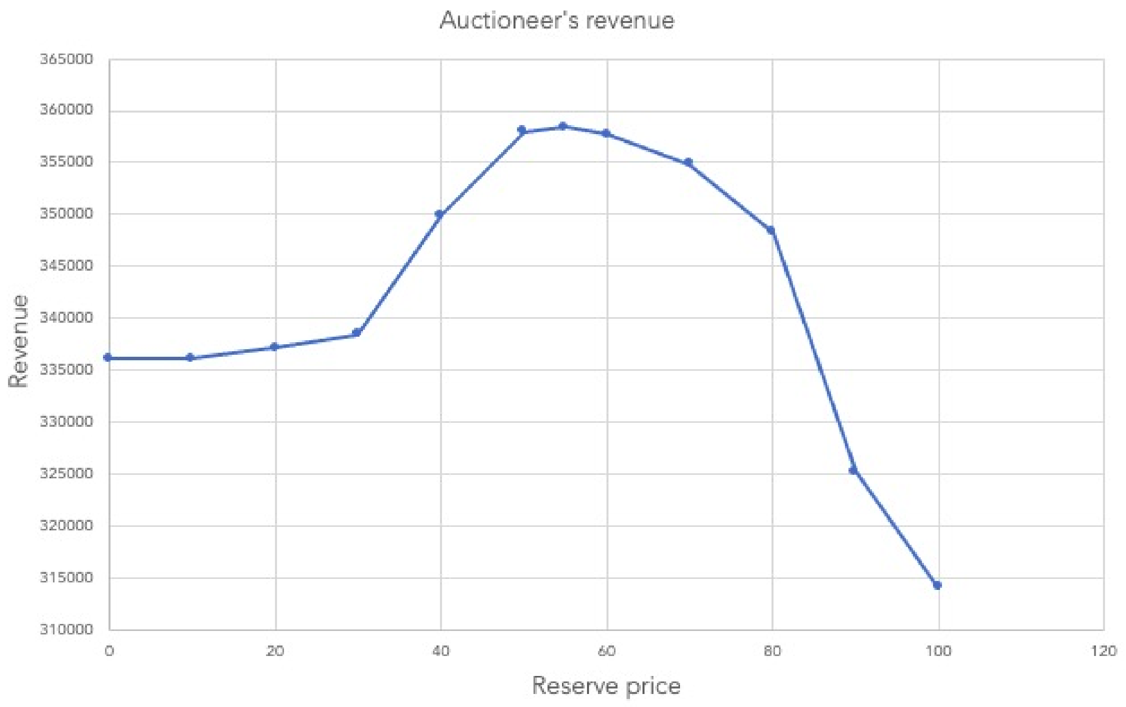
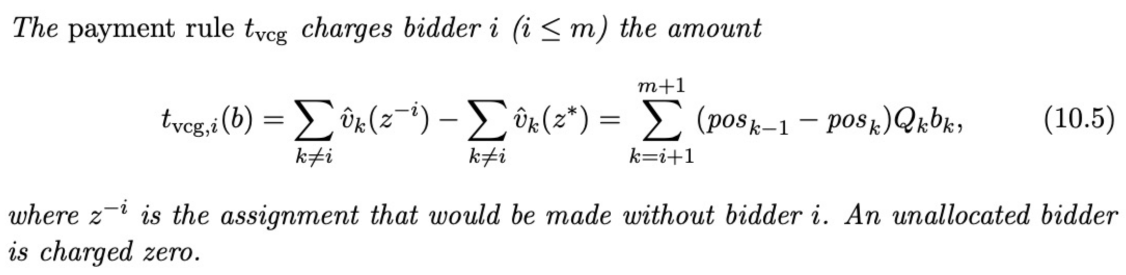


Figure 3.1.1: Auctioneer's revenue.

3.2 Implementation of the VCG slot

In a VCG auction each agent has to pay the negative consequence that it has imposed on other agents. The equation of a VCG payment rule is illustrated in figure 3.2.1



The payment rule t_{vcg} *charges bidder i ($i \leq m$) the amount*

$$t_{\text{vcg},i}(b) = \sum_{k \neq i} \hat{v}_k(z^{-i}) - \sum_{k \neq i} \hat{v}_k(z^*) = \sum_{k=i+1}^{m+1} (pos_{k-1} - pos_k) Q_k b_k, \qquad (10.5)$$

where z^{-i} is the assignment that would be made without bidder i. An unallocated bidder is charged zero.

Figure 3.2.1: VCG payment rule.

The box below contains the code to implement this equation.

```
            if (i < numSlots - 1) {
                int id = bids.get(i).getFirst();
                int steps = alloc - (id + 1);
                int sum = 0;
                for (int k = id; k < (id + steps); k++) {
                    int deltaclick = slotClicks.get(k) - slotClicks.get(k + 1);
                    sum = sum + deltaclick;
                    totalPayment = totalPayment + sum * bids.get(k + 1).getSecond();}
                perClickPayments.add((int) Math.round(totalPayment / (double) slotClicks.get(i)));
            }
            else {
                totalPayment = slotClicks.get(i) * firstUnallocatedBid;
                perClickPayments.add((int) Math.round(totalPayment / (double) slotClicks.get(i)));
            }
        }
        return perClickPayments;
```

3.3 A comparison between the VCG and GSP at r=0

If the population of agents is constituted by only truthful agents, then the auctioneer's revenue is halved in a VCG auction compared to the GSP model.

Table 3.1: Auctioneer's revenue in VCG and in GSP

maxperms=1 seed=2 numiters=50 reserve=0		
GSP	VCG	Difference
479345,240	262845,760	216499,48

3.4 A comparison between the VCG, GSP and hybrid at r=0

If the population of agents is formed by only BB agents, then the auctioneer is better off with a GSP auction. Its revenue decreases when switching to the VCG model in the middle of the simulation. The worst scenario in terms of auctioneer's revenue appears when the auction rules are following the VCG model.

Table 3.2: Auctioneer's revenue in VCG and in GSP

maxperms=1 seed=2 numiters=50 reserve=0		
GSP	VCG	Switch
336160,020	157931,960	244539,040

Chapter 4

Competition

4.1 BudgetPIBO agent

This special agent does not have to make predictions regarding the number of clicks because it knows the distribution's law of the number of clicks. I thought this would come as an asset to the agent and reduce uncertainty. The box below presents the steps to construct the array of clicks in the function expectedUtilities() and when formulating the bid.

```
        ArrayList<Integer> iterations = history.getSlotClicks(t-1);;
        ArrayList<Integer> numclicks = new ArrayList<>();
        int topslot = (int) (Math.round(35*Math.cos(Math.PI*t/24) + 40));
        for (int j=0; j<iterations.size();++j) {
            int element;
            if (j == 0) {
                element = topslot;
                numclicks.add(element);}
            else {
                int previous=j-1;
                element = (int) (Math.round(Math.pow(0.7,previous) * topslot));
                numclicks.add(element);}}
```

4.2 Win the competition

The BudgetPIBO agent wins, by far, the first competition against a population of Truthful Agents! The table 4.1 reports the results.

Table 4.1: Competition against Truthful Agents

maxperms=1 seed=2 numiters=50 reserve=10		
Agent	Type	Avg. Utility
0	TruthfulAgent	24489
1	TruthfulAgent	19269
2	TruthfulAgent	24638
3	TruthfulAgent	25532
4	BudgetPIBO	36111

Unfortunately, the performance against a population of BB Agents is less brilliant. The table 4.2 reports the results.

Table 4.2: Competition against BB Agents

maxperms=1 seed=2 numiters=50 reserve=10		
Agent	Type	Avg. Utility
0	BBAgentPIBO	47473
1	BBAgentPIBO	41453
2	BBAgentPIBO	47526
3	BBAgentPIBO	48572
4	BudgetPIBO	37411