COMPGW02: Web Economics
Assignment 1 - Part A

Sergiu Tripon
Department of Computer Science
University College London
Gower Street, London, WC1E
sergiu.tripon.15@ucl.ac.uk

1 REAL-TIME BIDDING [1]

Question 1

In a second price auction, with a single and indivisible item, the dominant strategy is truthful bidding. That is, each bidder will bid their true value of the item. In this context, we know the advertiser's true value for each user click on the ad and we also know the click-through rate on the ad. Therefore, her truthful bidding b_i will be:

$$b_i = r_i$$

where b_i is advertiser i's truthful bidding on the bid request and r_i is her true value for each user click on her ad.

Let's take advertisers p and i with true values r_p and r_i, respectively. If both p and i use the truth telling bidding strategy, in a second price ad auction, they will always bid their true value of the item, and therefore, if their value of the item is the same, they will bid the same price for each auction:

$$b_p = r_p$$
$$b_i = r_i$$

In order to prove this we need to verify that if p bids $b_p = r_p$, there is no other strategy that would improve advertiser p's payoff, regardless of the bidding strategy employed by any other advertiser. Below, we will verify two hypothetical scenarios where p increases or decreases their bid.

In both hypothetical scenarios, it is important to consider that the value that p bids only has an impact on whether p wins or loses the auction. However, it does not have any impact on what p will pay in the event where p wins, as the amount to be paid is determined by the other bids, more specifically by the largest of the other bids.

Firstly, let's assume that instead of advertiser p bidding r_p, they actually bid $b_p > r_p$. This only has an impact on p's payoff, if p would lose with bid $b_p = r_p$ but p would win with bid $b_p > r_p$. For this to happen, the other highest bid p_k must be between $b_p = r_p$ and $b_p > r_p$. In this case, p's payoff would be maximum $r_p - b_p \leq 0$, which does not increase p's payoff.

Secondly, let's assume that instead of advertiser p bidding r_p, they actually bid $b_p < r_p$. This only has an impact on p's payoff, if p would win with bid $b_p = r_p$ but p would lose with bid $b_p < r_p$. For this to happen, the other highest bid p_k must be between $b_p = r_p$ and $b_p < r_p$. In this case, p's payoff would be 0 because p loses, which does not increase p's payoff.

Figure 1 provides a visual representation of this proof.

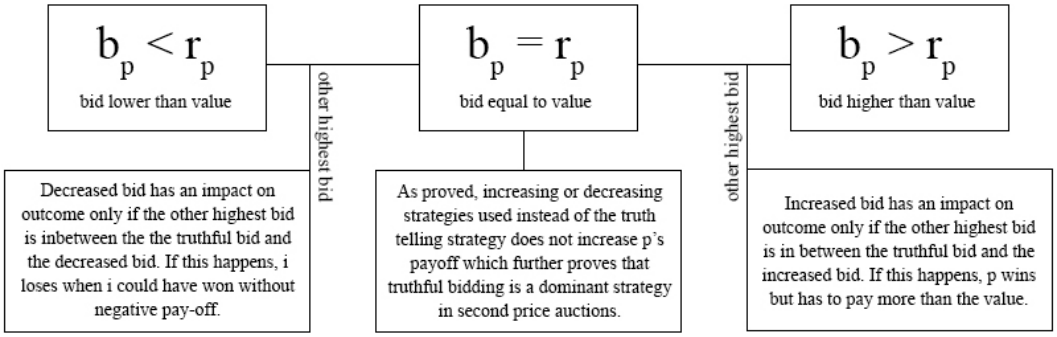

Figure 1: Proof of why truthful bidding is a dominant strategy in second price auctions.

Question 2

The right choice of changing her truth-telling bid price in order to maximise the ad click number is to keep the bid price level.

In *Question 1*, it has been proved that any deviation from the advertiser's true value-based bid $b_i = r_i$ would not improve her payoff and is not optimal. On the other hand, the advertiser bidding their true value is optimal. This makes truthful bidding a dominant strategy in second price auctions, with a single and indivisible item.

It is not optimal for advertiser i to bid more than her true value, because it will only help her to win the slot if the second highest bidder's bid is at least equal to her true value. In this scenario, advertiser i will not gain anything, but may lose something.

It is not optimal for advertiser i to bid less than her true value, because her bid does not have an impact on the price she has to pay if she wins the slot. In this scenario, advertiser i's lower bid may make her lose the slot even thought it would've been advantageous for her to win it as the price is lower than her true value.

2 SPONSORED SEARCH AUCTION [1] [2]

The contents of my *Auction.py* file can be found, below.

```
# Auction.py

#!/usr/bin/python
import Bid, Slot

class Auction:
  'This class represents an auction of multiple ad slots to multiple
      advertisers'
  query = ""
  bids = []

  def __init__(self, term, bids1=[]):
    self.query = term

    for b in bids1:
      j=0
      print(len(self.bids))
      while j<len(self.bids) and float(b.value)
          <float(self.bids[j].value):
        j+=1
      self.bids.insert(j,b)

  '''
```

```
This method accepts a Vector of slots and fills it with the results
of a VCG auction. The competition for those slots is specified in the
    bids Vector.
@param slots a Vector of Slots, which (on entry) specifies only the
    clickThruRates
and (on exit) also specifies the name of the bidder who won that slot,
the price said bidder must pay,
and the expected profit for the bidder.
'''

def executeVCG(self,slots):
  # TODO: implement this method
  # print ("executeVCG: To be implemented")

  price = 0.0
  lowClickThruRate = 0.0

  # for loop as many times as number of slots, descending >> if 4
      slots, it will loop: 3, 2, 1, 0
  for bid in range(len(slots)-1, -1, -1):
    # if bid no. is smaller than number of bids, continue
    if bid < len(self.bids):
      # if current bid no. + 1 is smaller than number of bids, do
          calculations
      if bid+1 < len(self.bids):
        # calculate price: bid amount * (current slot's clickThruRate
            - lowClickThruRate)
        price += self.bids[bid+1].value * (slots[bid].clickThruRate -
            lowClickThruRate)
      # assign current bidder's price
      slots[bid].price = price
      # assign current bidder's name
      slots[bid].bidder = self.bids[bid].name
      # calculate and assign current bidder's profit: (bid amount *
          clickThruRate) - price
      slots[bid].profit = (self.bids[bid].value *
          slots[bid].clickThruRate) - slots[bid].price
    # if bid no. is larger than number of bids, assign zero values
    elif bid > len(self.bids):
      # assign zero value to current bidder's price
      slots[bid].price = 0
      # assign zero value to current bidder's name
      slots[bid].bidder = 0
      # assign zero value to current bidder's profit
      slots[bid].profit = 0
    # assign current slot's clickThruRate to lowClickThruRate
    lowClickThruRate = slots[bid].clickThruRate
```

References

[1] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. Chap. 9, pp. 254–257.

[2] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. Chap. 15, pp. 449–452.