Mechanism Design for Auctions

Vadim Ovtchinnikov

September 30, 2009

Abstract

Mechanism design, the science of building economic systems, has been impressively successful. It has been used to create the auctions in which the FCC sold frequency licenses for over 100 billion dollars. Used to design the mechanisms for selling internet advertising, an industry with revenues of 8 billion dollars annually. In this paper, four interesting auction designs have been coded in Python and AMPL. Combinations of each design are simulated across different environments, measuring efficiency, revenue, surplus, and running time. Conclusions are drawn as to which combinations of these designs are best suited for different situations.

1 Introduction

Auctions are mechanisms to trade goods between sellers and buyers by offering commodities up for a bid, taking the bids, and then selling them to the winner of the auction. Historically the word auction is derived from Latin augere and means "to increase" or "augment", it lives up to this name because the mechanism is designed to increase the flow of wealth through the auction. Like the latin language, it has a long history, recorded as early as in 500 [4] B.C . With the beginning of the Internet era, this form of commodity trading increased in popularity, but also in complexity. The idea of one- good; multiple buyer auctions of a good being sold at the strike of the hammer, must be extended when the desire is for example selling advertisements spots on TV channel.

Suppose that some seller wants to reach out to small group of customers. If he advertises in a niche TV program his chances selling the product will increase. For example, if a company sells golf clubs, it wants to place commercial ads in a sports related program, that attracts golfers. This reaches a large population of potential buyers. Other types of advertisers preferences can be expressed in the number of impressions one spot attracts on a certain TV program. For example the advertiser is selling milk and wants to reach a broad audience. We can make this even more complicated if we include combinatorial preferences, for example, when the buyer is willing to buy both spot A and B, but if he can not get them both, he would like C instead.

A real world example of a complicated auction, the simultaneous multiple round auction, was first used by the Federal Communications Commission (FCC) in 1995 and copied around the world. The SMR has been used to sell over US $100 billion in spectrum licenses. Keyword auctions in Internet advertising, an industry that has annual revenue of US $8 billion, is being used in display advertising as well. Many companies have begun to employ mechanism design, notably Google and Yahoo!

2 Discussion

We implemented four interesting designs. Resource Allocation Design *(RAD)* [1]. This design can be compared to a sealed-bid auction. The second one is Combinatorial Clock *(CC)* [2], where sellers specify their valuations for the product and the auction runs automatically with robo-bidders with a fixed price increment. The price is increased until the demand for the products is less or equal to the supply. The third design is CC+WD, where WD stands for "Winner Determination" and is run in conjunction with CC. It takes in a collection of "remembered bids" instead of an environment, and generates a collection of "accepted bids" which imply an allocation. It is used in conjunction with CC to allocate excess supply to increase revenue. The final design is a combination of RAD+CC+WD, where after RAD the starting price is determined, and from there CC is run in conjunction with WD.

These four methods are implemented in the program language AMPL [5], an algebraic modeling language used for optimization programming. AMPL can be used to solve linear-, non-linear- and integer-programming, discrete variables is necessary when we are dealing with auctions. Generation of the environment was also made for two different cases: First a one-seller, multiple-bidders environment and second the TV advertisement environment. In order to generate statistics, we randomly generated 100 different environments with Python [3], and ran the four designs for each and every draw. Data of $Efficiency$, $Revenue$, $SellersSurplus$, $BuyersSurplus$ and $NumberofIterations$ were analyzed by creating plots and averages.

(1) One seller, multiple buyers Results show that RAD+CC+WD is the best choice in this case. It has the highest $Efficiency$, $Revenue$, $SellersSurplus$ and $BuyersSurplus$ *(See defenition, (3.5))*. The computational burden is also low.

(2) TV advertisement environment In this environment, RAD gives the best $Efficiency$, but only thanks to high $BuyersSurplus$, $SellersSurplus$ is 0%. Sellers have no profit, and therefore RAD can be omitted from the analysis. The CC+WD and RAD+CC+WD have both good values, but RAD+CC+WD has better values per iteration.

For these two different environments, the conclusion is that the RAD+CC+WD auction design gives highest values per iteration, and therefore preferable.

3 METHODS

3.1 The Supply Side

There are M commodities: $i = 1, ..., M$. Each commodity offers Q^i units at a reservation price ρ^i. Each unit of commodity i generates g^i units of an attribute, a vector with additional information that can differ commodities from one another. For TV environments can the numbers of impressions/viewers for a certain spot i be stored in g^i.

3.2 The Demand Side

There are J buyers: $j = 1, ..., J$. Each buyer is characterized by a budget B^j, a payoff function $u^j = \sum \beta_i^j x_i^j + \mu^j$ (bidders cash distribution, $u^j = B^j$), and constraints $\alpha_i^j x_i^j \leq y_i^j$, $\sum_i \alpha_i^j x_i^j \leq Z^j$, where x_i^j is the amount j receives of i, y maximum number of commodities buyer j wants from

buyer i, Z maximum number of commodities buyer j wants from all the sellers i, μ^j is the amount of cash that j has left, and α_i^j is a parameter that can be used to introduce buyers that values commodities with additional information ($g^i \neq 1$).

3.3 Environments

An environment is

$\{M, Q^1, ..., Q^M, \rho^1, ..., \rho^M, g^1, ..., g^M, J, B^1, ..., B^J, \beta_1^1, ..., \beta_M^J, \alpha_1^1, ..., \alpha_M^J, y_M^1, ..., y_M^J, Z^1, ..., Z^J\}$. For a given (M, J) there are $3M + J(3M + 2)$ parameters to generate.

3.3.1 Environment Generation

A reasonably general framework was programmed to handle different kind of environments. The generalization was made for two possibilities.

(1) very simple We should generate environments with 1 seller of 1 item with $Q^1 = 2, g^1 = 1$ and $\rho^1 \in [0, 100]$, 5 buyers with $y_1^j = 1, Z^j = 1, \alpha_1^j = 1, B^j = 500$ and $\beta_1^j \in [75, 200]$

(2) TV ad environments Two types of buyers: Spot buyer values spots from different programs the same and Impression buyer have a yen for having a high numbers of viewers/imoressions per spot.

4 programs: For each program i,

Spots: $Q^i \in [60, 110]$
Impressions/spot: $g^i \in [30, 80]$
Reserve price/spot: $\rho^i \in [300, 800]$

6 buyers: 3 spot buyers, 3 impression buyers

Spot buyers: For each of these j

Budget: $B^j \in [25000, 40000]$
Max/spot: $b^j \in [450, 900]$
Max spots: $Z^j \in [45, 70]$
Note that: $\beta_i^j = b^j, y_i^j = Z^j, \alpha_i^j = 1, \forall i.$

Impression buyers: For each of these j

Budget: $B^j \in [25000, 40000]$
Max/impression: $b^j \in [8.5, 12]$
Max impressions: $Z^j \in [2000, 5000]$
Note that:$\beta_i^j = b^j g^i, \alpha_i^j = g^i, y_i^j = Z^j, \forall i.$

3.4 Maximum Surplus Calculation

We assume utilities are quasi-linear in money. That is, we have a general equilibrium setup. In this case, given an environment, the maximum surplus calculation is:

$$\max_x \sum_j \sum_i (\beta_i^j - \rho^i) x_i^j \tag{1}$$

subject to

$$\alpha_i^j x_i^j \leq y_i^j, \forall i, j \tag{2}$$

$$\sum_i \alpha_i^j x_i^j \leq Z^j, \forall j \tag{3}$$

$$\sum_i x_i^j \leq Q^i, \forall i \tag{4}$$

$$x_i^j \text{ integer}, \forall i, j \tag{5}$$

By assuming that the buyers have infinite amount of money we obtain the upper bound for gains from trades, Maximum Surplus. This is a logical way of obtaining an upper bound for the maximization problem, done in one shot ($P_i = \rho_i$ for $\forall i$) and without disturbing the number of items each buyers wants (2)&(3). Notice that this is equivalent to omitting the budget constraints from RAD calculation (3.6.1). Efficiencies can be computed using Maximum Surplus as a reference value.

3.5 Outputs

Sellers Surplus Indicates how well sellers did in the auction, how much they gained from trading their commodities. For example, if one seller have a reserve price, $\rho = 5$ on an object. That is the least price he will consider selling the object for. A winner is determined by the auction at the price $P = 7$. Then $SellersSurplus = P - \rho = 7 - 5 = 2$.

Buyers Surplus The buyer, in the previous example placed a bid of $\beta = 12$, his gain from trade will be the $SellersSurplus = \beta - P = 12 - 7 = 5$.

Gains From Trade $SellersSurplus + BuyersSurplus = GainsFromTrade$, the total gain of both sides. In the example it would be, $SellersSurplus + BuyersSurplus = 2 + 5 = 7$.

Efficiency $GainsFromTrade/MaximumSurplus = Efficency$

Revenue Is the the price times allocation, how much money was paid in the auction. In the example $Revenue = P[j, i] * x[j, i] = 7 * 1 = 7$

Number of Iterations Every time we are forced to solve an maximization problem counts as one iteration.

3.6 Auction Processes

3.6.1 RAD

RAD is a sealed-bid auction.

RAD takes $\{M, Q^1, ..., Q^M, \rho^1, ..., \rho^M, g^1, ..., g^M, J, B^1, ..., B^J, \beta_1^1, ..., \beta_M^J, y_M^1, ..., y_M^J, Z^1, ..., Z^J, \alpha_1^1, ..., \alpha_M^J\}$ and produces an allocation $x^* \in \Re^{JM}$ and a set of prices $P^* \in \Re^M$

The allocations We choose x^* to solve:

$$\max_x \sum_{j=1}^{J} \sum_{i=1}^{M} [\beta_i^j - \rho^i] x_i^j \tag{6}$$

subject to

$$\alpha_i^j x_i^j \leq y_i^j \quad \forall i, j \tag{7}$$

$$\sum_{i=1}^{M} \alpha_i^j x_i^j \leq Z^j \quad \forall j \tag{8}$$

$$\sum_{i=1}^{M} \beta_i^j x_i^j \leq B^j \quad \forall j \tag{9}$$

$$\sum_{j=1}^{J} x_i^j \leq Q^i \quad \forall i \tag{10}$$

$$x_i^j \text{ integer}, \forall i, j \tag{11}$$

The prices Prices, P^*, are computed after x^* is computed in the previous section. For each program i,

$$A^i = \min_{j \in W^i} \beta_i^j \tag{12}$$

$$D^i = \max_{j \in L^i} \beta_i^j \tag{13}$$

$$C^i = \max\{D^i, \rho^i\} \tag{14}$$

$$P_i^* = \min\{A^i, C^i\} \tag{15}$$

$$\text{where } j \in W^i \text{ if } x_i^{*j} > 0 \text{ and} \tag{16}$$

$$j \in L^i \text{ if } \alpha_i^j x_i^{*j} < y^j \beta_i^j, \sum_i \alpha_i^j x_i^{*j} < Z^j, \text{ and } \sum_i \beta_i^j x_i^{*j} < B^j \tag{17}$$

Notes

- The RAD allocation problem differs from the maximum surplus problem because of (9).
- In pricing, the issue is which j to min or max across.

  For (12) include all j such that $X^{*j} > 0$.

  For (13) there exist individuals who would have been willing to have more i. These do not include those with the binding constraints $\alpha_i^j x_i^{*j} = y_i^j$, $\sum_i \alpha_i^j x_i^{*j} = Z^j$, and $\sum_i \beta_i^j x_i^* = B^j$.

Example 1.1

Simple environment, homogenous commodities	
One seller ($M = 1$)	Five buyers ($J = 5$)
$Q^1 = 2$ $\rho^1 = 80$	$\beta = [70, 90, 85, 84, 72]^T$ $B^j = 100$ for all $j = 1, 2, ..., 5$
RAD Auction	
$Revenue = 80 * 1 + 80 * 1 = 160$ $BuyersSurplus = (90 - 80) + (85 - 80) = 15$ $SellersSurplus = (80 - 80) + (80 - 80) = 0$	

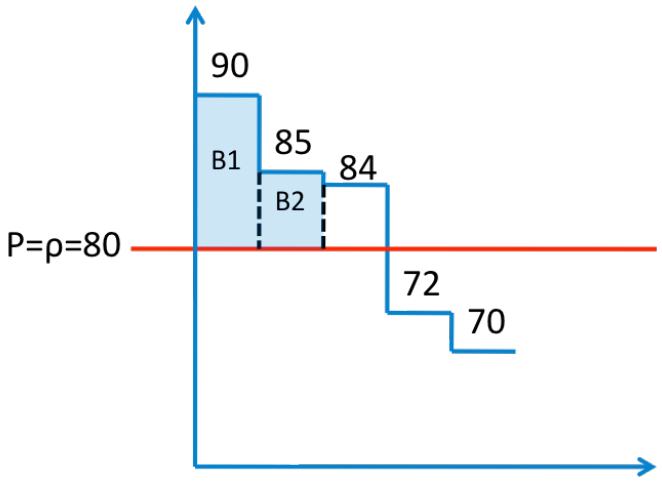


Figure 1: RAD

Equation *(6)* will be fulfilled if the first commodity is sold to the highest bidder and the second one to the next highest bidder, to increase the buyers gains from trade *(see Figure 1)*. The RAD is designed to be a one shot auction were the commodities are sold for the reservation price ρ, so the seller is not gaining from the trades.

3.7 CC with robo-bidders

CC is an iterative auction. This is treated as a sealed-bid auction by using robo-bidders in the CC: that is, let each buyer express their environment and then bid for them. This turns CC into a sealed-bid version.

CC takes $\{M, Q^1, ..., Q^M, \rho^1, ..., \rho^M, J, B^1, ..., B^J, \beta_1^1, ..., \beta_M^J, y_M^1, ..., y_M^J, Z^1, ..., Z^J\}$ and produces an allocation $x^* \in \Re^{JM}$ and a set of prices $P^* \in \Re^M$

CC proceeds in rounds, $\tau = 1, ...$ where for round τ

- CC posts prices $P(\tau)$
- Each buyer submits a bid $q^j(\tau)$. (See the next subsection.)
- We save the bidder's bid in this round τ, $(P(\tau), q^j(\tau))$, to the data base.

- If $\sum_j q_i^j(\tau) \leq Q^i$ for all i then go to WD.

  Note: If WD is not running then CC stops at this time. The price is $P(\tau) = P^*$ and the allocation is $x^{*j} = q^j(\tau)$.

- $P^i(\tau + 1) = (1 + d)P^i(\tau)$ if $\sum_j q_i^j(\tau) > Q^i$
- $P^i(\tau + 1) = P^i(\tau)$ if $\sum_j q_i^j(\tau) \leq Q^i$
- Go to next round $\tau + 1$ of CC

3.8 Robo-bidders

When CC requests bids from the bidders, here is the problem the computer solves for each j. Assume that bidders are straight-forward, that is, they report honestly and will want to bid honestly.

The jth bidders bid is: $(B^j, \beta_1^j, ..., \beta_M^j, \alpha_1^j, ..., \alpha_M^j, y_M^j, ..., y_M^j, Z^j)$.

Given the prices $P(\tau)$ we choose $q^j(\tau)$ to solve:

$$\max_{q^j} \sum_{i=1}^{M} [\beta_i^j - P^i(\tau)] q_i^j \tag{18}$$

subject to

$$\alpha_i^j q_i^j \leq y_i^j \quad \forall i \tag{19}$$

$$\sum_{i=1}^{M} \alpha_i^j q_i^j \leq Z^j \tag{20}$$

$$\sum_{i=1}^{M} P_i(\tau) q_i^j \leq B^j \tag{21}$$

$$q_i^j \text{ integer}, \forall i \tag{22}$$

Example 1.2

Simple environment, homogenous commodities	
One seller ($M = 1$)	Five buyers ($J = 5$)
$Q^1 = 2$ $\rho^1 = 80$	$\beta = [70, 90, 85, 84, 72]^T$ $B^j = 100$ for all $j = 1, 2, ..., 5$
CC Auction, with price increment 4	
$GainsFromTrade = (B + S)/(B + S + C) = (2 + 8)/(2 + 8 + 5) = 2/3 < 1$ $Revenue = 88 * 1 = 88$ $BuyersSurplus = (90 - 88) = 2$ $SellersSurplus = (88 - 80) = 8$	

In the CC auction, equation *(18)* must be satisfied. That means maximizing the surplus for each and every buyer such that constraints *(19)-(22)* are fulfilled and run the auction until the demand are less or equal the the capacity. The figure illustrates how the price is increased two times, until it reaches $P = 88$, we have the demand less or equal to the capacity, one commodity is allocated to the highest bidder, the other one remains unsold *(see Figure 2)*. The area C is not contributing to the gains from trade.

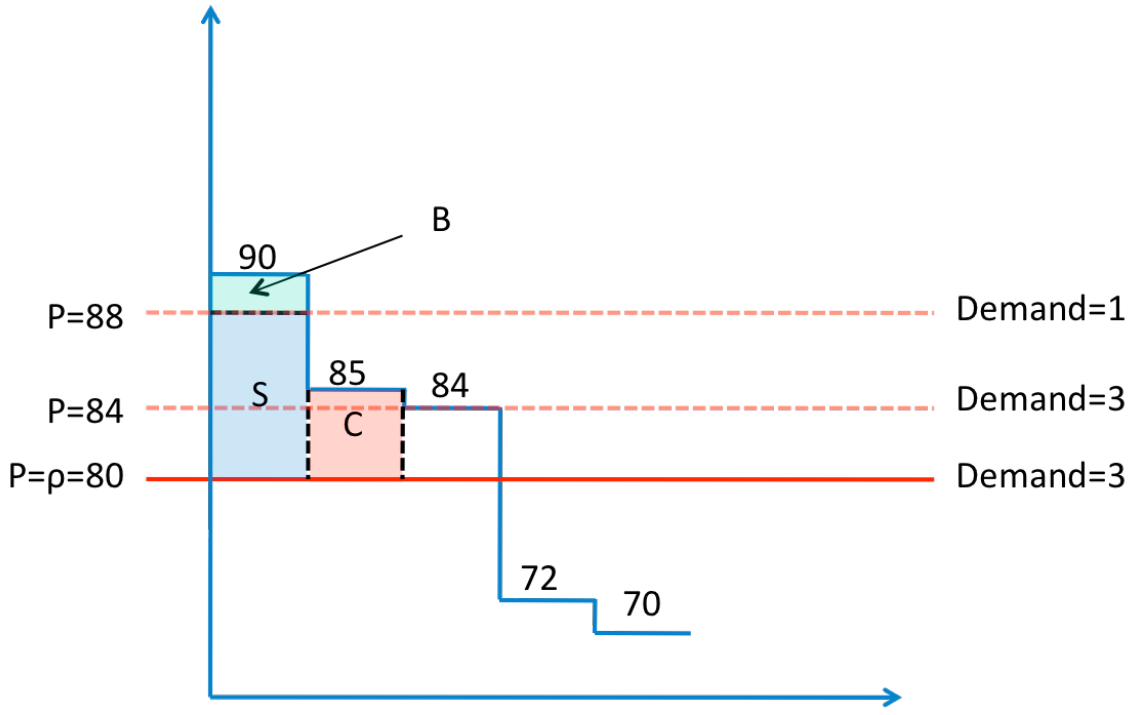


Figure 2: CC

3.9 WD

WD is a sealed bid auction run in conjunction with CC. It takes in a collection of "remembered bids", instead of an environment, and generates a collection of "accepted bids" which imply an allocation. It is used in conjunction with CC to allocate excess supply to increase revenue.

Note that if $t = 1$ is the last CC round, then we just skip WD and use the CC allocation for the outcome of the auction.

A remembered bid is $[\tau, P_1^j(\tau), ..., P_M^j(\tau), x_1^j(\tau), ..., x_M^j(\tau)]$. This is to be thought of as what j bids (i.e., was willing to buy) in round τ of the clock when prices were $P(\tau)$. For $\tau = 1, ..., t$ (τ is the set of all the rounds and t is the last round), there will be a max of tJ bids in the remembered set. Let $x^j = 0$ for those rounds in which j did not bid, then there will be t bids for each j. Let $\delta^j(\tau) = 1$ if j's bid is accepted from iteration τ. Otherwise, $\delta^j(\tau) = 0$.

WD solves the following problem: choose $[\delta^j(\tau), y_i^j(\tau)], \forall i, j, \tau$ to maximize

$$\max_{z,\delta} \sum_{j,i,\tau} (P^i(\tau) - \rho^i) z^j_{i,\tau} \tag{23}$$

subject to

$$\sum_{j,\tau} z^j_{i,\tau} \leq Q^i, \forall i \tag{24}$$

$$z^j_{i,\tau} \leq x^j_i(\tau)\delta^j(\tau) \tag{25}$$

$$\sum_{\tau \leq t} \delta^j(\tau) \leq 1 \tag{26}$$

$$\delta^j(\tau) \in \{0, 1\} \tag{27}$$

$$z^j_{i,\tau} \text{ is an integer.} \tag{28}$$

$$\text{and if } x^j(t) \neq 0, \text{ then } \delta^j(\tau) = 0, \forall \tau < t. \tag{29}$$

The idea is to take all past bids (including the current winning ones from CC) and pick those that would maximize seller's surplus, to be able to partially fill with past bids: this is the reason for the z and constraint (25). Using only one past bid per bidder: this is constraint (26). And, not using past bids of current winners: this is constraint (29).

Once WD is computed, the decision point is to go back to CC or stop. Stopping rule is if no winner from CC at t has been displaced in WD. Let y^* be the solution to the above problem.

If $z^{j*}_{i,t} = x^j_i(t)$ for all i, for all j such that $x^j_i(t) \neq 0$ for some i, then stop. The winning allocation is z^* and the winners pay $P^i(\tau)$ for $z^{j*}_i(\tau)$.

Otherwise,

let $P^i(t+1) = P^i(t)(1+d)$ if there is a j such that $z^{j*}_{i,t} < x^j_i(t)$, and

let $P^i(t+1) = P^i(t)$ if $z^{j*}_{i,t} = x^j_i(t)$ for all j such that $x^j_i(t) > 0$.

Then return to CC. Start there with round $t+1$ and $P(t+1)$.

Example 1.3

Notice that in Example 1.2, the CC design fails to allocate all the commodities which is not beneficial for the seller. We introduce a Winner Determination block, which maximizes the sellers surplus by using past bids as environment.

Simple environment, homogenous commodities	
One seller ($M = 1$)	Five buyers ($J = 5$)
$Q^1 = 2$ $\rho^1 = 80$	$\beta = [70, 90, 85, 84, 72]^T$ $B^j = 100$ for all $j = 1, 2, ..., 5$
CC+WD Auction	
$GainsFromTrade = 1$ $Revenue = 88 * 1 + 84 * 1 = 172$ $BuyersSurplus = (90 - 88) + (85 - 84) = 3$ $SellersSurplus = (88 - 80) + (84 - 80) = 12$	

There is three past rounds to chose the allocation from. Round number three, ($t = 3$) had a bid for one of the commodity for the highest price, P= 88. By accepting this bid, we maximize sellers

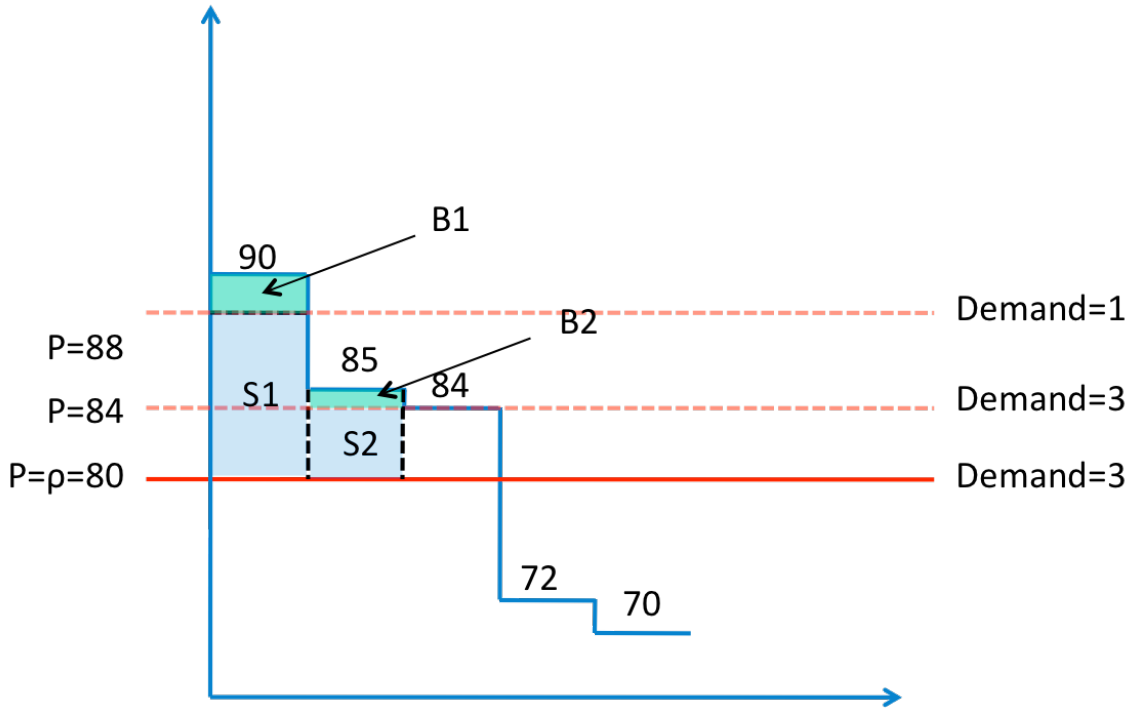


Figure 3: CC+WD

surplus. Second item had a bid in second round, ($\tau = 2$). Sellers surplus will be maximized if we fixate that item to next-highest or third-highest bidder *(see Figure 3)*. Hopefully the item will go to next highest-bidder, because it maximizes even the buyers surplus and do not generate a side-market. In the computer implementation of the problem the second item will be allocated to next highest bidder, but only thanks to the indexation of the bid-vector. To really create a fair auction WD-block can be modified to also maximize the sellers surplus, but this increase the computational burden.

3.10 Results

For each round, the following computations were made: RAD alone, CC alone, CC+WD and RAD+CC+WD. The results were calculated and displayed for 100 rounds, *Revenue*, *SellersSurplus* and *BuyersSurplus*, are divided with *MaximumSurplus (3.4)*. *Revenue* is defined as price times allocation, *SellersSurplus* and *BuyersSurplus* indicates how well seller respectively buyers did in the auction, their profit from trades. By summing them together we get gains from trades. If we divide it by maximum gains from trade we obtain efficiency.

The plots *(see Appendix (6))* displays the distribution function $p(k)$ =(number of objective less or equal to k) $\div$ (number of rounds), where $k = 0, 1, ..., 100$ represents the percentage.

We can see that the curves statistically dominate one another by being shifted to the right without overlaps and therefore can be ranked as superior across the entire distribution.

Average for Simple environment, after 100 rounds					
Designs	$Efficiency$	$Revenue$	$BuyersSurplus$	$SellersSurplus$	$NumberofIterations$
RAD	14.0%	12.8%	14.0%	0.0%	1.00
CC	3.5%	3.8%	1.9%	1.5%	12.67
CC+WD	72.4%	68.2%	38.4%	34.0%	13.59
RAD+CC+WD	74.8%	69.8%	39.2%	35.6%	7.97

Average for TV advertisement environment, after 100 rounds					
Designs	$Efficiency$	$Revenue$	$BuyersSurplus$	$SellersSurplus$	$NumberofIterations$
RAD	73.2%	133.9%	73.2%	0.0%	1.00
CC	44.2%	106.6%	18.1%	26.0%	9.74
CC+WD	68.7%	175.7%	21.2%	47.5%	10.46
RAD+CC+WD	66.5%	155.9%	20.6%	45.9%	3.35

The number of iterations is the number of times we need to solve an optimization problem in order to find an allocation.

3.11 Conclusions

We can clearly see from the averages that for the Simple environment, RAD+CC+WD is clearly the best design, it's domination across the hole distribution is also evident from the plots.

For the complex, TV advertisement environment we can see that RAD design has the highest $Efficiency$, but only because of the high $BuyersSurplus$, in the long run sellers want show up for an auction where there surplus is 0%, therefore this design can be left out from the analysis. We can notice that CC design did allot better with the complex environment, compared to the simple environment, and in fewer iterations. Less number of iterations is true for all the designs, which is an interesting phenomena. The best averages are obtained from the CC+WD design, is also the dominating one from the distribution plots (if we omit the RAD design). But RAD+CC+WD, achieves almost as good averages but in less iterations, so if the computational burden is a problem this design is 3 times faster then the RAD+CC+WD design.

The question of whether there exist even better computer implemented auction designs is still open.

The next step to modify this designs is to introduce a double maximization in the WD block, as mentioned in the *Example 1.3*. We need to prefer the maximization across the past bids not only for the $SellersSurplus$, but also for the $BuyersSurplus$. It may increase the computational burden rapidity. To obtain the difference in $Efficiency$-changed per number of iterations, we need to run the tests.

4 Acknowledgments

Thanks to Prof. John Ledyard and co- mentor Guilherme de Freitas at California Institute of Technology (USA) and Dr. Henrik Sandberg at Royal Institue of Technology (Sweden) for the mentorship in this project.

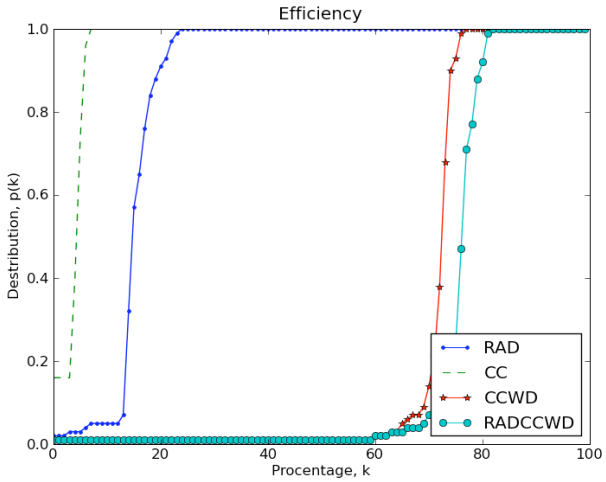
5 Author Information

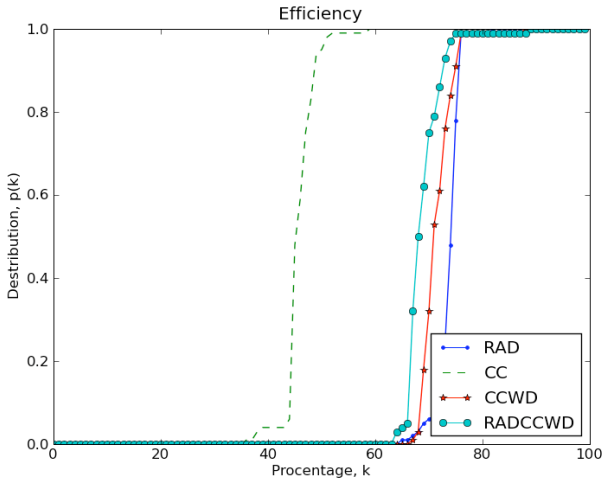
Vadim Ovtchinnikov

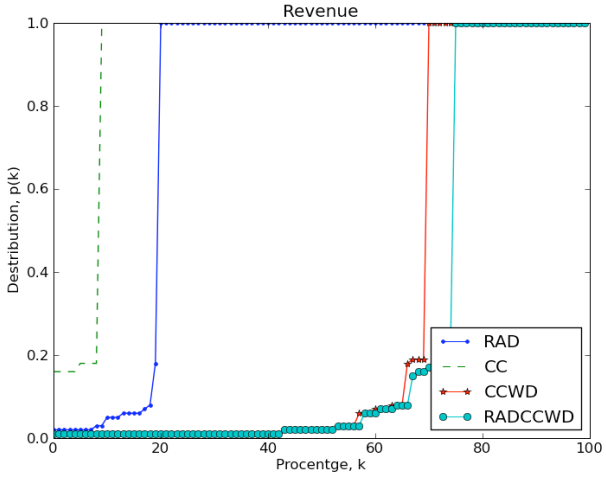
Undergraduate student, Master Engineering Physics
Royal Institute of Technology, Stockholm
vadimo@kth.se

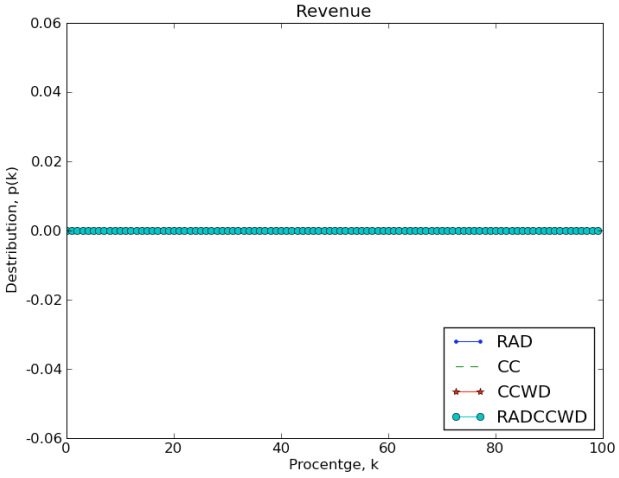
6 Appendix

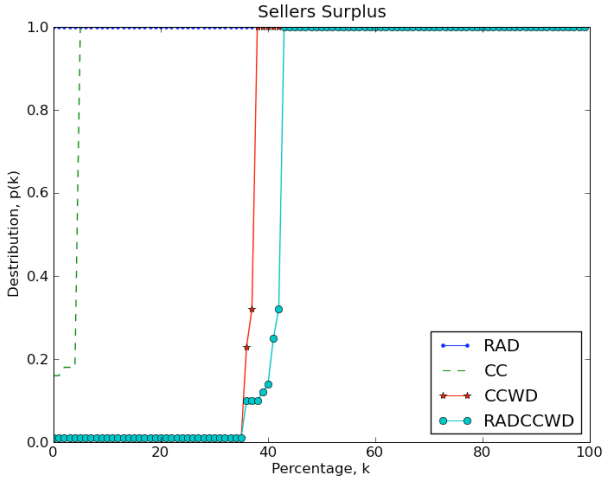
Table 1: Simple (left column) & TV advertisement (right column)- environment, distribution

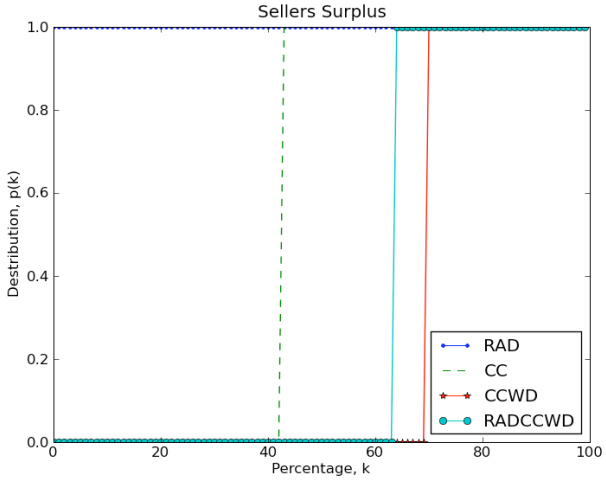












References

[1] John Ledyard Christine DeMartini, Anthony Kwasnica and David Porter. A new and improved design for multi- object iterative auctions. *Management Science*, 51(3):419–434, 1990.

[2] Anil Roopnarinec David Portera, Stephen Rassentia and Vernon Smitha. Combinatorial auction design. *Proceedings of the National Academy of Sciences*, 100(19):11153–11157, 2003.

[3] Michael H. Goldwasser. *Object-Oriented Programming in Python*. Prentice Hall, 2008.

[4] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[5] David M. Gay Robert Fourer and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Brooks, Cole Publishing Company, 2003.