Allocation of slots in position auctions

Value of an agent $i = P_i \left( \hat{\theta}_i \cdot \theta_i \right) = v_i (a, \theta_i)$

where $a = (a_1, \ldots, a_n)$ is the allocation, $a_i$ is the slot allocated to $i$.

Pick allocations $a^* \in \arg\max \sum_{i \in N} v_i (a, \theta_i)$ efficient allocation

Claim: An allocation of slots is efficient iff it is a rank-by-expected revenue mechanism.

Proof sketch: Maximizing the weighted sum problem. Sum is maximized when maximum weight is put on maximum value.

The slot allocation problem is a sorting problem - hence computationally tractable.

Allocation decision is done, need payments to make it DSIC.

Natural candidate: VCG [used in Facebook]

Note: actual implementation in practice might be different. Here we discuss only an abstract notion of how it can be done.

VCG in position auction

Given bids $(b_1, \ldots, b_n)$ [note, $\hat{\theta}_i$ : reported type and $b_i$ are same]

WLOG ordered such that $\hat{\theta}_1, b_1 \succ \hat{\theta}_2, b_2 \succ \cdots \succ \hat{\theta}_n, b_n$

so allocation $a^*$ is s.t. $a^*_i = i$. 

* define $a^*_i \in \arg\max_{a \in A} \sum_{j \neq i} v_j(a, \theta_j)$

note: allocations of the agents after $i$, i.e., $i+1$ to $n$ get one slot better.

hence $p_{i}^{\text{VCG}}(b) = \sum_{j \neq i} v_j(a^*_i, \theta_j) - \sum_{j \neq i} v_j(a^*, \theta_j)$

$= \sum_{j = i}^{n-1} p_j(\hat{e}_{j+1} b_{j+1}) - \sum_{j = i}^{n-1} p_{j+1}(\hat{e}_{j+1} b_{j+1})$

$= \sum_{j = i}^{n-1} (p_j - p_{j+1})(\hat{e}_{j+1} b_{j+1})$, $\forall i = 1, \ldots, n-1$

$p_n^{\text{VCG}}(b) = 0$.

This is the total expected payment. To convert this to the pay-per-click:

$$\frac{1}{p_i \hat{e}_i} p_i^{\text{VCG}}(b).$$