CSE 6319 Notes 3: Mechanism Design (Part 2)

(Last updated 3/12/24 3:21 PM)

3.F. MULTI-UNIT AUCTIONS

Uniform-Price for Budgeted Bidders (R 9.2)

m = number of units

Besides private valuation v_i, each buyer has a public budget B_i

Demand of bidder i at price p: $$D_i(p) = \begin{cases} \min\left\{\lfloor \frac{B_i}{p} \rfloor, m\right\} & \text{if } p < v_i \\ 0 & \text{if } p > v_i \end{cases}$$

Utility is still $v_i - p$

Not DSIC (due to "ambiguity" for $p = v_i$, see R p. 115 Example 9.1) - *demand reduction* for reporting a lower v_i

Clinching for Budgeted Bidders (R 9.3; `PAPERSONE/dobzinski.pdf`)

Price "gradually" rises to point where demand for all but one buyer (with largest residual demand) is below remaining supply.

If total demand exceeds supply, allocate one unit to largest-residual-demand buyer (and continue).
Otherwise, allocate remaining supply at current price.

DSIC

$m = 25$

i	B_i	v_i
1	100	24
2	200	22
3	300	20

s	p	B_1	$D_1(p)$	B_2	$D_2(p)$	B_3	$D_3(p)$
25	5	100	$\boxed{20}$	200	$\boxed{40}$	300	60
			<----- 20+40 > s - 1 ---->			Highest demand	
25	12.5	100	8	200	16	300	24 (1 at 12.5)
			<----- 8+16 $\leq s$ - 1 ---->			Highest demand	
24	12.5	100	$\boxed{8}$	200	$\boxed{16}$	287.5	23
24	12.6	100	7	200	15	287.5	22 (1 at 12.6)
23	12.6	100	7	200	15	274.9	21 (1 at 12.6)

22	12.6	100	$\boxed{7}$	200	$\boxed{15}$	262.3	20
22	13.4	100	7	200	14	262.3	19 (1 at 13.4)
21	13.4	100	$\boxed{7}$	200	$\boxed{14}$	248.9	17
21	14.3	100	6	200	13	248.9	17 (1 at 14.3)
20	14.3	100	6	200	13	234.6	16 (1 at 14.3)
19	14.3	100	$\boxed{6}$	200	$\boxed{13}$	220.3	15
19	15.4	100	6	200	12	220.3	14 (1 at 15.4)
18	15.4	100	$\boxed{6}$	200	$\boxed{12}$	204.9	13
18	16.7	100	5	200	11	204.9	12 (1 at 16.7)
17	16.7	100	5	200	11	188.2	11 (1 at 16.7)
16	16.7	100	5	200	11 (1 at 16.7)	171.5	10
15	16.7	100	$\boxed{5}$	183.3	$\boxed{10}$	171.5	10
15	17.3	100	5	183.3	10 (1 at 17.3)	171.5	9
14	17.3	100	$\boxed{5}$	166	$\boxed{9}$	171.5	9
14	18.5	100	5	166	8	171.5	9 (1 at 18.5)
13	18.5	100	$\boxed{5}$	166	$\boxed{8}$	153	8
13	19.2	100	5	166	8 (1 at 19.2)	153	7
12	19.2	100	$\boxed{5}$	146.8	$\boxed{7}$	153	7
12	20	100	5 (5 at 20)	146.8	7 (7 at 20)	153	0

Clinching with Descending Valuations (`PAPERSONE/ausubel.pdf`; R Problem 9.2)

Ascending auction with private valuations, but payments are based on prices where demand falls below remaining supply

Agent *i* valuations (for individual units) are downward-sloping $v_{i1} \geq v_{i2} \geq \ldots \geq v_{ik}$ for auction with *k* units (replaces budgets)

$m = 5$

Marginal Values	1st Unit	2nd Unit	3rd Unit
Bidder A	123	113	103
Bidder B	75	5	3
Bidder C	125	125	49
Bidder D	85	65	7
Bidder E	45	25	5
Bidder F	49	9	3

Price	Demand						Sum
	A	B	C	D	E	F	
0	3	3	3	3	3	3	18
3	3	2	3	3	3	2	16
5	3	1	3	3	2	2	14
7	3	1	3	2	2	2	13
9	3	1	3	2	2	1	12
25	3	1	3	2	1	1	11
45	3	1	3	2	0	1	10
49	3	1	2	2	0	0	8
65	3	$\underline{1}$	$\underline{2}$	$\underline{1}$	0	0	7

```
Demand of 4  A gets 1st unit at 65
```

```
Price     Demand       Sum
          A B C D E F
75        3 0 2 1 0 0   6
          3 + 1 = 4 ⇒ C gets 1st unit at 75
          2 + 1 = 3 ⇒ A gets 2nd unit at 75
85        3 0 2 0 0 0   5
          2 ⇒ A gets 3rd unit at 85
          3 ⇒ C gets 2nd unit at 85
```

Nisan survey (`PAPERSONE/nisanAmd.pdf`, Algorithmic Mechanism Design: Through the lens of Multi-unit auctions)

- We have m identical indivisible units of a single good to be allocated among n strategic players also called *bidders*.
- Each bidder i has a privately known *valuation function* $v_i : \{0, ..., m\} \to \Re^+$, where $v_i(k)$ is the value that this bidder has for receiving k units of the good. We assume that $v_i(0) = 0$ and free disposal: $v_i(k) \leq v_i(k+1)$ (for all $0 \leq k < m$).
- Our goal (as the auctioneer) is to allocate the units among the bidders in a way that optimizes *social welfare*: each bidder i gets m_i units and we aim to maximize $\sum_i v_i(m_i)$, where the feasibility constraint is that $\sum_i m_i \leq m$.

Representations

Bidding Languages

1. **Single Minded Bids:** This language allows only representing "step functions", valuations of the form $v(k) = 0$ for $k < k^*$ and $v(k) = w^*$ for $k \geq k^*$. Clearly to represent such a valuation we only need to specify two numbers: k^* and w^*. Clearly, also, this is a very limited class of valuations.
2. **Step Functions:** This language allows specifying an arbitrary sequence of pairs $(k_1, w_1), (k_2, w_2), \ldots, (k_t, w_t)$ with $0 < k_1 < k_2 \cdots < k_t$ and $0 < w_1 < w_2 < \cdots < w_t$. In this formalism $v(k) = w_j$ for the maximum j such that $k \geq k_j$. Thus, for example, the bid $((2, 7), (5, 23))$ would give a value of \$0 to 0 or 1 items, a value of \$7 to 2, 3, or 4 items, and a value of \$23 to 5 or more items. Every valuation can be represented this way, but most valuations will take length m. Simple ones – ones that have only a few "steps" – will be succinctly represented.
3. **Piece-Wise Linear:** This language allows specifying a sequence of marginal values rather than of values. Specifically, a sequence $(k_1, p_1), (k_2, p_2), \ldots, (k_t, p_t)$ with $0 < k_1 < k_2 \cdots < k_t$ and $p_j \geq 0$ for all j. In this formalism p_j is the marginal value of item k for $k_j \leq k < k_{j+1}$. Thus $v(k) = \sum_{l=1}^{k} u_l$ with $u_l = p_j$ for the largest j such that $l \geq k_j$. In this representation, the bid $((2, 7), (5, 23))$ would give a value of \$7 to 1 item, \$14 to 2 items, \$37 to 3 items, \$60 to 4 items, and \$83 to 5 or more items.

Value Queries - any $v_i(k)$ instance in polynomial time in the representation length

Communication Queries - values are not provided exhaustively; instead queries are posed to the bidders

(Intractable) General Allocation Algorithm (DP, section 4.1) - O(nm^2) time

The Allocation Problem

Input: The sequence of valuations of the players: $v_1, \ldots, v_n$.
Output: An allocation $m_1, \ldots, m_n$ with $\sum_{i=1}^n m_i \le m$ that maximizes $\sum_{i=1}^n v_i(m_i)$.

1. Fill the $(n+1)*(m+1)$ table s, where $s(i,k)$ is the maximum value achievable by allocating k items among the first i bidders:
   (a) For all $0 \le k \le m$: $s(0,k) = 0$
   (b) For all $0 \le i \le n$: $s(i,0) = 0$
   (c) For $0 < i \le n$ and $0 < k \le m$: $s(i,k) = max_{0 \le j \le k}[v_i(j) + s(i-1, k-j)]$
2. The total value of the optimal allocation is now stored in $s(n,m)$
3. To calculate the m_i's themselves, start with $k = m$ and for $i = n$ down to 1 do:
   (a) Let m_i be the value of j that achieved $s(i,k) = v_i(j) + s(i-1, k-j)$
   (b) $k = k - m_i$

Tractable Downward Sloping Valuations via Market Equilibrium and Multiple Binary Searches

$v_i(k+1) - v_i(k) \le v_i(k) - v_i(k-1)$ for all bidders i and number of items $1 \le k \le m-1$

Algorithmically, given a potential price p, we can calculate players' demands using binary search to find the point where the marginal value decreases below p. This allows us to calculate the total demand for a price p, determining whether it is too low or too high, and thus search for the right price p using binary search. For clarity of exposition we will assume below that all values of items are distinct, $v_i(k) \ne v_{i'}(k')$ whenever $(i,k) \ne (i',k')$.

Allocation Algorithm for Downward Sloping Valuations

1. Using binary search, find a clearing price p in the range $[0, V]$, where $V = max_i[v_i(1)]$:
   (a) For each $1 \le i \le n$, use binary search over the range $\{0, 1, ..., m\}$, to find m_i such that $v_i(m_i) - v_i(m_{i-1}) \ge p > v_i(m_{i+1}) - v_i(m_i)$
   (b) If $\sum_i m_i > m$ then p is too low; if $\sum_i m_i < m$ then p is too high, otherwise we have found the right p.
2. The optimal allocation is given by $(m_1, ..., m_n)$.

										k										
	$v_i(k)$	0		1		2		3		4		5		6		7		8
	1	0		50		100		150		190		220		240		250		260
			50		50		50		40		30		20		10		10	
i	2	0		70		120		170		210		245		275		300		310
			70		50		50		40		35		30		25		10	
	3	0		60		110		160		200		235		270		295		320
			60		50		50		40		35		35		25		25	

p	m_1	m_2	m_3	Σ
10	8	8	8	24
20	6	7	8	21
30	5	6	6	17
35	4	5	6	15
40	4	4	4	12
50	3	3	3	9
60	0	1	1	2
70	0	1	0	1

Intractability for value queries model

Theorem 3 *Every algorithm for the multi-unit allocation problem, in the value-queries model, needs to make at least $2m - 2$ queries for some input.*

Let us consider the simplest possible bidding language, that of Single Minded Bids. In this language each valuation v_i is represented as (k_i, p_i) with the meaning that for $k < k^*$ we have $v_i(k) = 0$ and for $k \geq k_i$ we have $v_i(k) = p_i$. The optimal allocation will thus need to choose some subset of the players $S \subseteq \{1 \ldots n\}$ which it would satisfy. This would give total value $\sum_{i \in S} p_i$ with the constraint that $\sum_{i \in S} k_i \leq m$. This optimization problem is a very well known problem called the knapsack problem, and it is one of a family of problems known to be "NP-complete". In particular, no computationally-efficient algorithm exists for solving the knapsack problem unless "P=NP", which is generally conjectured not to be the case (the standard reference is still Garey and Johnson [1979]). It follows that no computationally-efficient algorithm exists for our problem when the input valuations are presented in any of the bidding languages mentioned above all of which contain the single-minded one as special cases. We thus have:

Theorem 4 *Assuming $P \neq NP$, there is no polynomial-time algorithm for the multi-unit allocation problem, in any of the bidding language models.*

Communication Complexity Results

Theorem 5 *(Nisan and Segal [2006]) Every algorithm for the multi-unit allocation problem, in any query model, needs to ask queries whose total answer length is at least $m-1$ in the worst case.*

DP-based Approximation Scheme (with binary searches, from section 4.4)

At this point we seem to have very strong and general impossibility results for algorithms attempting to find the optimal allocation. However, it turns out that if we just slightly relax the optimality requirement and settle for "approximate optimality" then we are in much better shape. Specifically, an *approximation scheme* for an optimization problem is an algorithm that receives, in addition to the input of the optimization problem, a parameter ϵ which specifies how close to optimal do we want to solution to be. For a maximization optimization problem (like our allocation problem) this would mean a solution whose value is at least $1-\epsilon$ times the value of the optimal solution. It turns out that this is possible to do, computationally-efficiently, for our problem using dynamic programming.

4. Generality (p. 11 of CSE 5311 Notes 11
`https://ranger.uta.edu/~weems/NOTES5311/NEWNOTES/notes11.pdf`)

Approximation Algorithm - achieve max/min ratio in $O\left(n^k\right)$ time (k fixed)

Approximation Scheme - flexible ratio 1 + ε in $O\left(f\left(n,\varepsilon\right)\right)$

Polynomial-time Approximation Scheme - $O\left(n^{f\left(\varepsilon\right)}\right)$

Fully PTAS - $O\left(n^k\left(\frac{1}{\varepsilon}\right)^l\right)$ time

The basic idea is that there exists an optimal algorithm whose running time is polynomial in the *range of values*. (This is a different algorithm than the one presented in section 4.1 whose running time was polynomial in the number of items m.) Our approximation algorithm will thus truncate all values $v_i(k)$ so that they become small integer multiples of some δ and run this optimal algorithm on the truncated values. Choosing $\delta = \epsilon V/n$ where $V = max_i[v_i(m)]$ strikes the following balance: on one hand, all values are truncated to an integer multiple $w\delta$ for an integer $0 \leq w \leq n/\epsilon$, and so the range of the total value is at most the sum of n such terms, i.e. at most n^2/ϵ. On the other hand, the total additive error that we make is at most $n\delta \leq \epsilon V$. Since V is certainly a lower bound on the total value of the optimal allocation this implies that the fraction of total value that we loose is at most $(\epsilon V)/V = \epsilon$ as required.

The allocation algorithm among the truncated valuations uses dynamic programming, and fills a $(n+1)*(W+1)$-size table K, where $W = n^2/\epsilon$. The entry $K(i,w)$ holds the minimum number of items that, when allocated optimally between the first i players, yields total value of at least $w\delta$. Each entry in the table can be computed efficiently from the previous ones, and once the table is filled we can recover the actual allocation.
(Keller, H. et.al. *Knapsack Problems*, Springer, 2004 is very useful on DP for maximizing value or minimizing number of units)

1. Fix $\delta = \epsilon V/n$ where $V = max_i[v_i(m)]$, and fix $W = n^2/\epsilon$.
2. Fill an $(n+1)*(W+1)$ table K, where $K(i,w)$ holds the minimum number of items that, when allocated optimally between the first i players, yields total value of at least $w\delta$:
   (a) For all $0 \le i \le n$: $K(i,0) = 0$
   (b) For all $w = 1, \ldots, W$: $K(0,w) = \infty$
   (c) For all $i = 1, \ldots, n$ and $w = 1, \ldots, W$ compute $K(i,w) = min_{0 \le j \le w} val(j)$, for $val(j)$ defined by:
      i. Use binary search to find minimum number of items k such that $v_i(k) \ge j\delta$
      ii. $val(j) = k + K(i-1, w-j)$
3. Let w be the maximum index such that $K(n,w) \le m$
4. For i going from n down to 1:
   (a) Let j be so that $K(i,w) = val(j)$ (as computed above)
   (b) Let m_i be the minimum value of k such that $v_i(k) \ge j\delta$
   (c) Let w = w - j

Our algorithm's running time is dominated by the time required to fill the table K which is of size $n \times n^2/\epsilon$ where filling each entry in the table requires going over all possible n/ϵ values j and for each performing a binary search that takes $O(\log m)$ queries. Thus the total running time is polynomial in the input size as well as in the precision parameter ϵ which often called a fully polynomial time approximation scheme. For most realistic optimization purposes such an arbitrarily good approximation is essentially as good as an optimal solution. We have thus obtained:

Theorem 6 *There exists an algorithm for the approximate multi-unit allocation problem (in any of the models we considered) whose running time is polynomial in n, $\log m$, and ϵ^{-1}. It produces an allocation $(m_1, \ldots, m_n)$ that satisfies $\sum_i v_i(m_i) \ge (1-\epsilon)\sum_i v_i(opt_i)$, where $(opt_1, \ldots, opt_n)$ is the optimal allocation.*

Payments, Incentives, and Mechanisms

Direct Revelation Mechanism

Based on Revelation Principle, assume truthful $v_i(k)$ values are provided

VCG Mechanism

1. Each Player reports a valuation $\tilde{v}_i$.
2. The mechanism chooses the allocation $(m_1, ..., m_n)$ that maximizes $\sum_j \tilde{v}_j(m_j)$ and outputs it.
3. For each player i:
   - (a) The mechanism finds the allocation $(m'_1, ..., m'_n)$ that maximizes $\sum_{j \neq i} \tilde{v}_j(m'_j)$.
   - (b) Player i pays $p_i = \sum_{j \neq i} \tilde{v}_j(m'_j) - \sum_{j \neq i} \tilde{v}_j(m_j)$.

"The basic difficulty in the field of Algorithmic Mechanism Design is to *simultaneously* achieve both incentive compatibility and computational efficiency."

Approximation and Incentives

"desire a . . . mechanism that for every given $\in > 0$ produces an allocation whose total value is at least 1 - $\in$ fraction of the optimal one"

Efficient?

Truthful?

Truncation may lose truthfulness (p. 25-26). (issue with earlier DP-based Approximation Scheme)

Maximum in Range Mechanisms

Efficiency issue when maximizing $\sum_i v_i(m_i)$ over all possible allocations $(m_1, ..., m_n)$

Instead, maximize over a range of possible allocations that are fixed in advance

Assume m is an integer multiple of n^2

Approximation Mechanism

1. Split the m items into n^2 equi-sized bundles of size m/n^2 each
2. Run the optimal VCG mechanism with the items being the n^2 bundles, using the general allocation algorithm from section 4.1

Theorem 8 *(Dobzinski and Nisan [2007]) There exists a truthful mechanism for the 2-approximate multi-unit allocation problem (in any of the models we considered) whose running time is polynomial in n and $\log m$. It produces an allocation $(m_1, ..., m_n)$ that satisfies $\sum_i v_i(m_i) \geq (\sum_i v_i(opt_i))/2$, where $(opt_1, ..., opt_n)$ is the optimal allocation.*

Theorem 9 *(Dobzinski and Nisan [2007]) There is no Maximum-in-Range mechanism for approximate multi-unit allocation (in any query model) whose running time is polynomial in n and $\log m$ and always produces an allocation $(m_1, ..., m_n)$ that satisfies $\sum_i v_i(m_i) > (\sum_i v_i(opt_i))/2$, where $(opt_1, ..., opt_n)$ is the optimal allocation.*

Single Parameter Mechanisms

Single-Minded Bids - bidder *i* has value v_i for at least k_i units

Truthful mechanisms are simply those with:

1. **Monotonicity:** If a player wins with bid (k_i, v_i) then it will also win with any bid which offers at least as much money for at most as many items. I.e. i will still win if the other bidders do not change their bids and i changes his bid to some (k'_i, v'_i) with $k'_i \leq k_i$ and $v'_i \geq v_i$.
2. **Critical Payment:** The payment of a winning bid (k_i, v_i) is the smallest value needed in order to win k_i items, i.e. the infimum of v'_i such that (k_i, v'_i) is still a winning bid, when the other bidders do not change their bids.

Single-Minded Additive Approximation Mechanism

1. Let $\epsilon > 0$ be the required approximation level, and let δ be a truncation precision parameter
2. Truncate each v_i to $j\delta$ where $0 \leq j \leq n/\epsilon$ is an integer
3. Find the optimal allocation among the truncated v_i's, using the algorithm from section 4.4
4. Charge the critical payments

Single-Minded Approximation Mechanism:

1. Let $\epsilon > 0$ be the required approximation level.
2. For all $\delta = 2^t$ where t is an integer (possibly negative) with $\epsilon V/(2n^2) \leq \delta \leq V$, where $V = max_i v_i$ do:
   (a) Run the Additive Approximation Algorithm with precision parameter δ obtaining a "δ-scale allocation"
3. Choose the δ-scale allocation that achieved the highest value (as counted with the truncated values) to determine the final allocation
4. For each winner i in the final allocation: Find, using binary search, the lowest value that would still have bidder i win when the others' values are unchanged, and set this critical value as i's payment

Theorem 10 *(Briest et al. [2011]) There exists a truthful mechanism for the approximate multi-unit allocation problem among single-minded bidders whose running time is polynomial in n, $\log m$, and ϵ^{-1}. It produces an allocation $(m_1, ..., m_n)$ that satisfies $\sum_i v_i(m_i) \geq (1-\epsilon) \sum_i v_i(opt_i)$, where $(opt_1, ..., opt_n)$ is the optimal allocation.*

Multi-Parameter Mechanisms Beyond VCG?

Theorem 11 *(Lavi et al. [2003]) Every truthful mechanism for approximate multi-unit allocation with two players that always allocates all items, $m_1 + m_2 = m$, and for some fixed $\alpha > 1/2$ always satisfies $v_1(m_1)+v_2(m_2) \geq \alpha(v_1(opt_1)+v_2(opt_2))$, where (opt_1, opt_2) is the optimal allocation, requires at least m queries in the value queries model.*

Randomization

Maximum-in-Distributed-Range Mechanism

Theorem 12

3.G. Truthful Auctions in Win/Lose Settings (KP 15; N 9.3/9.5; R 7)

Win/Lose Allocation Settings (KP 15.2)

Single-minded for subset of items

- A set U of participants/bidders, where each has a private value v_i for "winning" (being selected) and obtains no value from losing;
- a set of feasible allocations (i.e., possible choices for the set of winning bidders) $\mathcal{L} \subset 2^U$. In a single-item auction, $\mathcal{L}$ contains all subsets of size at most 1 and in the communication channel example, $\mathcal{L} = \{S \subset U \mid \sum_{i \in S} w_i \leq C\}$.

Allocation Rule

Based on the vector of bids b_i, probability (often 0/1) for each bidder winning

$$\mathbf{x}(\mathbf{b}) = \underset{\omega \in \Omega}{\operatorname{argmax}} \sum_{i=1}^{n} b_i(\omega)$$

Payment Rule

Based on the vector of bids b_i, (expected) payment for each bidder

$$p_i(\mathbf{b}) = \underbrace{\max_{\omega \in \Omega} \sum_{j \neq i} b_j(\omega)}_{\text{without } i} - \underbrace{\sum_{j \neq i} b_j(\omega^*)}_{\text{with } i}$$

Social Surplus and the VCG Mechanism (R 7.2; KP 15.3; N 9.3.3)

Social Surplus: $\sum_{i \in U} \left(v_i \mathbb{1}_{\{i \text{ winner}\}} - p_i\right) + \sum_{i \in U} p_i = \sum_{i \in L^*} v_i$

- Ask each bidder to report his private value v_i (which he may or may not report truthfully). Assume that bidder i reports b_i.
- Choose as the winning set a feasible $L \in \mathcal{L}$ that maximizes $b(L)$, where $b(L) = \sum_{j \in L} b_j$. Call this winning set L^*.
- To compute payments, let $\mathcal{L}_i^+ = \{S | S \cup \{i\} \in \mathcal{L}\}$. Then i only wins if his bid b_i satisfies

$$b_i + \max_{L \in \mathcal{L}_i^+} b(L) \geq \max_{L \in \mathcal{L}^-} b(L), \qquad (15.1)$$

where $\mathcal{L}^-$ is the collection of sets in $\mathcal{L}$ that do *not* contain i. His payment is his **threshold bid**, the minimum b_i for which (15.1) holds; i.e.,

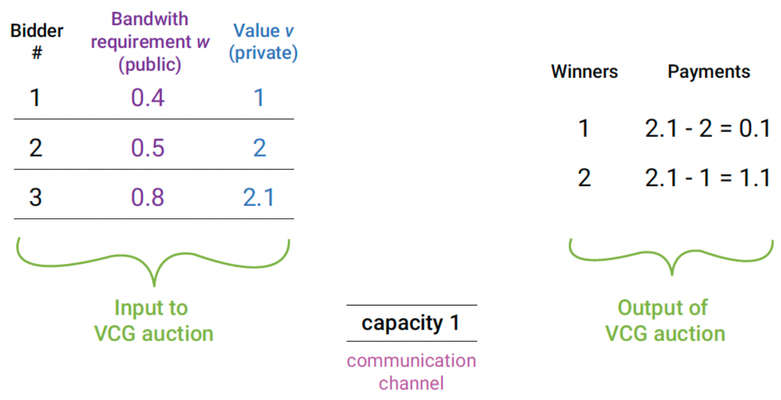
$$p_i = \max_{L \in \mathcal{L}^-} b(L) - \max_{L \in \mathcal{L}_i^+} b(L). \qquad (15.2)$$

Truthfulness and Individually Rational - KP Theorem 15.3.2

Envy-freeness - not guaranteed (KP p. 273)

Applications of VCG (KP 15.4; N 9.3.5)

Shared Communication Channel (knapsack auction restated; KP 15.4.1)



Bidder #	Bandwith requirement w (public)	Value v (private)
1	0.4	1
2	0.5	2
3	0.8	2.1

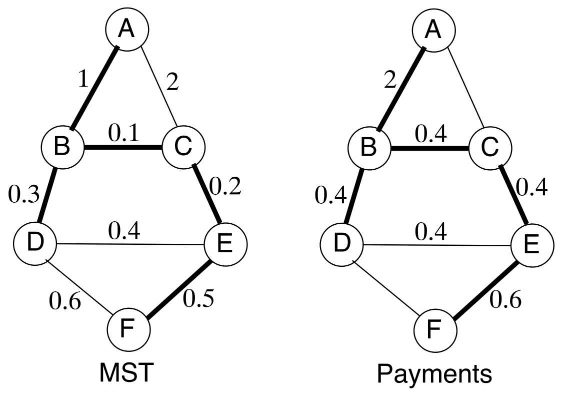
Winners	Payments
1	2.1 - 2 = 0.1
2	2.1 - 1 = 1.1

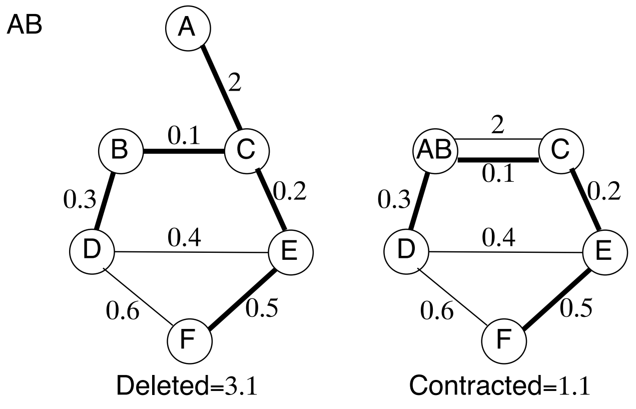
FIGURE 15.2. This figure illustrates the execution of the VCG algorithm on Example 15.1.3 (a shared communication channel) when $C = 1$ and there are three bidders with the given values and weights. In this example, $\mathcal{L} = \{\{1,2\},\{1\},\{2\},\{3\},\emptyset\}$. With the given values, the winning set selected is $\{1,2\}$. To compute, for example, bidder 1's payment, we observe that without bidder 1, the winning set is $\{3\}$ for a value of 2.1. Therefore the loss of value to other bidders due to bidder 1's presence is $2.1 - 2$.

Spanning Tree (KP 15.4.2)

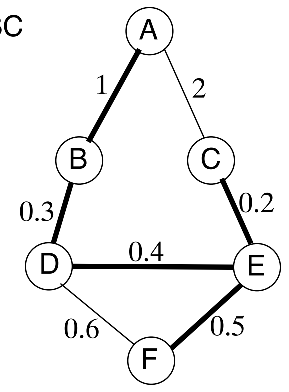
- Choose the MST T^* with respect to the reported costs.
- Pay each winning link owner ℓ his threshold bid, which is C if removing it disconnects the graph, and otherwise

  (cost of MST with ℓ deleted) $-$ (cost of MST with ℓ contracted),

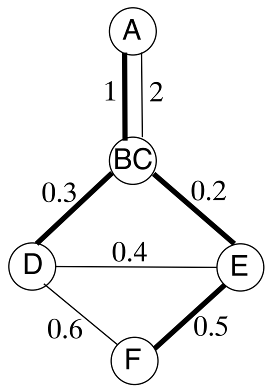




BC

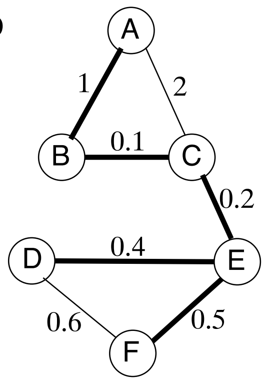


Deleted=2.4

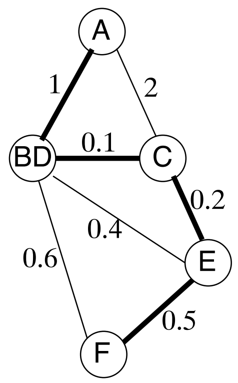


Contracted=2

BD

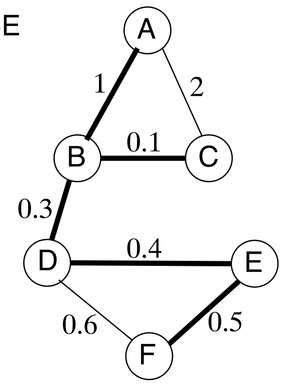


Deleted=2.2

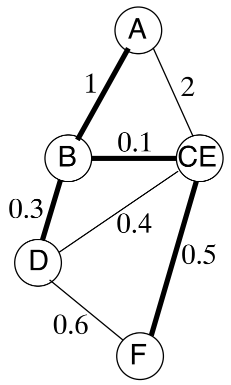


Contracted=1.8

CE

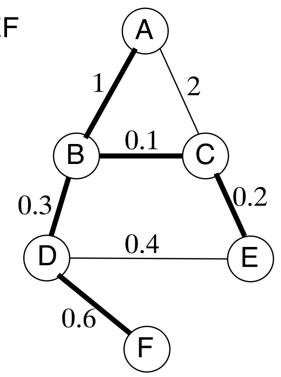


Deleted=2.3

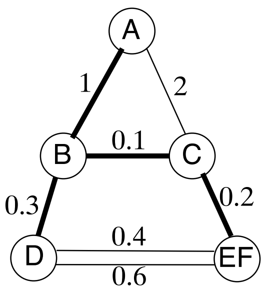


Contracted=1.9

EF

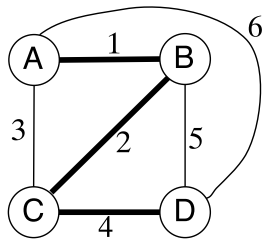


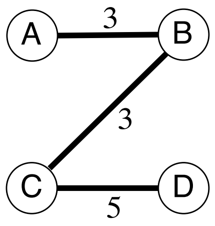
Deleted=2.2



Contracted=1.6

Another MST Example:

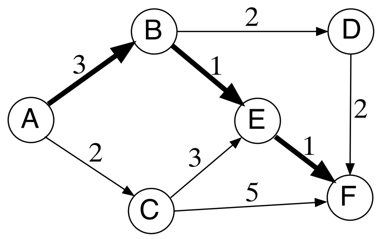




AB: 9 - 6 = 3 BC: 8 - 5 = 3 CD: 8 - 3 = 5

Aside:
`https://en.wikipedia.org/wiki/Steiner_tree_problem#Steiner_tree_in_graphs_and_variants`

Charging for Shortest Path (N 9.3.5.6)



Shortest-path length without edge (“deleted”) - cost with edge “contracted”

AB: 6 - 2 = 4 BE: 6 - 4 = 2 EF: 7 - 4 = 3

Public Project Issues with VCG (KP 15.4.3; N 9.3.5.5)

Budget Balance

Collusion

(Long term) value and (immediate) payment

Not “Envy-Free”

Definition 15.4.1. We say that a truthful mechanism in a win/lose setting is **envy-free** if, when bidding truthfully, each bidder prefers his own allocation and payment to that of any other bidder. That is, for every $\mathbf{v}$ and i, we have

$$\alpha_i[\mathbf{v}]v_i - p_i[\mathbf{v}] \geq \alpha_j[\mathbf{v}]v_i - p_j[\mathbf{v}].$$

Example 15.4.2. Suppose that the government is trying to decide whether or not to build a bridge from the mainland to a big island A or to a small island B. The cost of building a bridge is \$90 million. Island A has a million people, and each one values the bridge at \$100. Island B has five billionaires, and each one values the bridge at \$30 million. Running VCG will result in a bridge to island B and payments of 0. In this case, the people on island A will envy the outcome for the people on island B.

Other Resources:

"Lovely, but Lonely" paper - `PAPERSONE/ausubelMilgrom.pdf`

- low revenues
- non-monotonicity
- vulnerability to collusion
- vulnerability to aliases
- (revelation of too much private information)

Milgrom book (downloadable from UTA Library) -
`https://www-degruyter-com.ezproxy.uta.edu/document/doi/10.7312/milg17598/html`

3.H. Truthful Auctions in Win/Lose Settings (continued)

Sponsored Search Auctions (KP 15.5; N 28; R 2.6/3.5/5.3)

Slot Clickthrough Rates (probabilities?): $c_1 \geq c_2 \geq \ldots \geq c_k \geq 0$

Determine permutation π of valuations v_i (same as b_i) to maximize social surplus $\sum_{j=1}^{k} v_{\pi_j} c_j$

VCG auction:

- Each bidder is asked to submit a bid b_i representing the maximum he is willing to pay per click.
- The bidders are reordered so that their bids satisfy $b_1 \geq b_2 \geq \ldots$, and slot i is allocated to bidder i for $1 \leq i \leq k$.
- The participation of bidder i pushes each bidder $j > i$ from slot $j-1$ to slot j (with the convention that $c_{k+1} = 0$). Thus, i's participation imposes an expected cost of $b_j(c_{j-1} - c_j)$ on bidder j in one search (assuming that b_j is j's value for a click). The auctioneer then charges bidder i a price of $p_i(\mathbf{b})$ per click, chosen so that his expected payment in one search equals the total externality he imposes on other bidders; i.e.,

$$c_i p_i(\mathbf{b}) = \sum_{j=i+1}^{k+1} b_j(c_{j-1} - c_j). \tag{15.3}$$

In other words, bidder i's payment per click is then

$$p_i(\mathbf{b}) := \sum_{j=i+1}^{k+1} b_j \frac{c_{j-1} - c_j}{c_i}. \tag{15.4}$$

Examples (also see KP Figure 15.7):

k=3, n=4 c_1=0.48 c_2=0.24 c_3=0.16 c_4=0.0

v_1=\$4 v_2=\$3 v_3=\$2 v_3=\$1

$p_1 = v_2(c_1 - c_2)/c_1 + v_3(c_2 - c_3)/c_1 + v_4(c_3 - c_4)/c_1$
$= 3(0.48 - 0.24)/0.48 + 2(0.24 - 0.16)/0.48 + 1(0.16 - 0.0)/0.48 = \$2.16667$

$p_2 = v_3(c_2 - c_3)/c_2 + v_4(c_3 - c_4)/c_2 = 2(0.24 - 0.16)/0.24 + 1(0.16 - 0.0)/0.24 = \$1.33333$

$p_3 = v_4(c_3 - c_4)/c_3 = 1(0.16 - 0.0)/0.16 = \$1.00$

Another view of the VCG auction for sponsored search (KP 15.5.1)

The entire auction may be viewed as the sum of k single-slot auctions . . .

Leads to the entire auction being truthful (KP Lemma 15.5.2)

Generalized First Price - Historical Use . . .

Generalized Second Price (KP 15.5.2)

- Each advertiser interested in bidding on keyword K submits a bid b_i, indicating the price he is willing to pay per click.
- Each ad is ranked according to its bid b_i and ads allocated to slots in this order.
- Each winning advertiser pays the minimum bid needed to win the allocated slot. For example, if the advertisers are indexed according to the slot they are assigned to, with advertiser 1 assigned to the highest slot (slot 1), then advertiser i's payment p_i is

$$p_i = b_{i+1}.$$

Not truthful - KP Figure 15.9, but . . .

KP has an equilibrium strategy based on valuations and CTRs:

Lemma 15.5.3. *Consider n competing advertisers with values v_i sorted so that $v_1 \geq v_2 \geq \cdots \geq v_n$. Assuming truthful bidding in the VCG auction, from (15.4) we have that bidder i's price-per-click is*

$$p_i^{\mathrm{VCG}} = \sum_{j=i+1}^{k+1} v_j \frac{c_{j-1} - c_j}{c_i}. \tag{15.5}$$

Then, in GSP, it is a Nash equilibrium for these advertisers to bid $(b_1, \ldots, b_n)$ where $b_1 > p_1^{\mathrm{VCG}}$ and $b_i = p_{i-1}^{\mathrm{VCG}}$ for $i \geq 2$.

Reserve Prices and Yahoo! - R 5.3

Back to Revenue Maximization (KP 15.6)

Trips to the Moon - maximize the number of cost-sharing participants

```
Input T, n
while (true)
    share := T/n
    Ask the n remaining bidders "Will you pay share?"
    Input numberWilling
    if n == 0
        return 0
    if n == numberWilling
        return n
    n := numberWilling
```

$T = 100$

n:	10 ($10)	5 ($20)	5	
n:	20 ($5)	5 ($20)	2 ($50)	2
n:	10 ($10)	3 ($33.33)	0	

Revenue maximization *without* priors (KP 15.6.1; N 13.3)

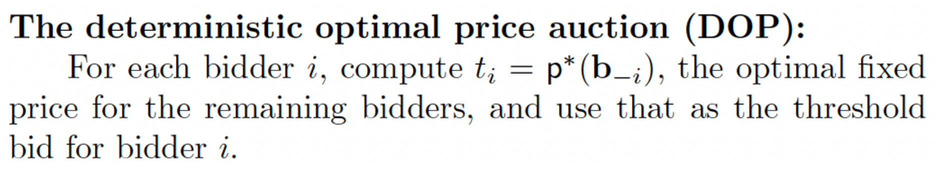
Digital goods to be sold at fixed price

Conceptually, order bids descending for obtaining *optimal fixed-price* and maximize price to fit first *i* bidders

i	1	2	**3**	4	5	6	7	8	9	10
b	10	8	**6**	4	2	1	1	1	1	1
Revenue	10	16	**18**	16	10	6	7	8	9	10

Using this "directly" is not truthful

Deterministic optimal price auction (DOP) is truthful, but poor revenue (unless appropriate "distributional" assumptions can be made)



The deterministic optimal price auction (DOP):
For each bidder i, compute $t_i = \mathsf{p}^*(\mathbf{b}_{-i})$, the optimal fixed price for the remaining bidders, and use that as the threshold bid for bidder i.

Example 15.6.4. Consider a group of bidders of which 11 bidders have value 100 and 1,001 bidders have value 1. The best fixed price is 100 - at that price 11 items can be sold for a total revenue of \$1,100. (The only plausible alternative is to sell to all 1,012 bidders at price \$1, which would result in a lower revenue.)

However, if we run the DOP auction on this bid vector, then for each bidder of value 100, the threshold price that will be used is \$1, whereas for each bidder of value 1, the threshold price is \$100, for a total revenue of only \$11!

All bidders bidding truthfully, solved optimally

i	1	2	...	**11**	12	13	...	1011	1012
b	100	100	100	**100**	1	1	1	1	1
Revenue	100	200	...	**1100**	12	13	...	1011	1012

DOP making decision for original bidders 1 ... 11 (removing any one of them)

i	1	2	...	10	11	12	...	1010	**1011**
b	100	100	100	100	1	1	1	1	**1**
Revenue	100	200	...	1000	11	12	...	1010	**1011**

Thus, these 11 are committed (individually) at a threshold of 1.

DOP making decision for original bidders 12 ... 1012 (removing any one of them)

i	1	2	...	**11**	12	13	...	1010	1011
b	100	100	100	**100**	1	1	1	1	1
Revenue	100	200	...	**1100**	12	13	...	1010	1011

Thus, these 1001 are out (individually) at a threshold of 100.

DOP revenue is 11

Revenue extraction (KP 15.6.2)

"Trips to the moon" technique is truthful and useful . . .

An approximately optimal auction - random sampling revenue extraction auction (KP 15.6.3; N 13.3.2)

Definition 15.6.7 (RSRE). The **random sampling revenue extraction auction** (RSRE) works as follows:

(1) Randomly partition the bids $\mathbf{b}$ into two groups by flipping a fair coin for each bidder and assigning her bid to $\mathbf{b}'$ or $\mathbf{b}''$.
(2) Compute the optimal fixed-price revenue $T' := \mathcal{R}^*(\mathbf{b}')$ and $T'' := \mathcal{R}^*(\mathbf{b}'')$.
(3) Run the revenue extractors: $\mathsf{pe}_{T'}$ on $\mathbf{b}''$ and $\mathsf{pe}_{T''}$ on $\mathbf{b}'$. Thus, the target revenue for $\mathbf{b}''$ is determined by $\mathbf{b}'$ and vice versa.

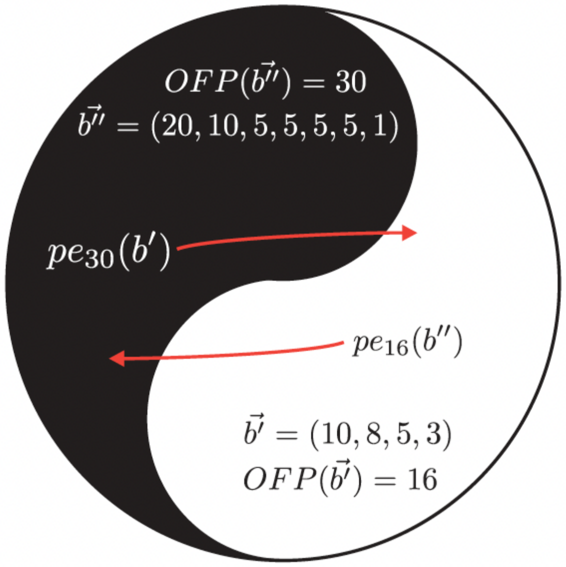


FIGURE 15.11. This figure illustrates a possible execution of the RSRE auction when the entire set of bids is $(20, 10, 10, 8, 5, 5, 5, 5, 5, 3, 1)$. Running the revenue extractor $\mathsf{pe}_{30}(\mathbf{b}')$ will not sell to anyone. Running the revenue extractor $\mathsf{pe}_{16}(\mathbf{b}'')$ will sell to the top six bidders at a price of 16/6.

THEOREM 15.6.9. *The random sampling revenue extraction (RSRE) auction is truthful and for all bid vectors* $\mathbf{b}$, *the expected revenue of RSRE is at least* $\mathcal{R}_2^*(\mathbf{b})/4$. *Thus, if bidders are truthful, this auction extracts at least* $\mathcal{R}_2^*(\mathbf{v})/4$ *in expectation.*