Payment Mechanism in Knapsack Auctions

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Mrinal Tak (130101049)

under the guidance of

G. Sajith

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

Contents

Chapter 1

Introduction

1.1 Organization of The Report

This chapter provides an introduction to the basics of Mechanism Design, Social Choice Function and Mechanism implementation. A flavor of VCG Mechanism is also provided to the reader. We then give an introduction to the Knapsack Auctions.

The rest of the chapters are organised as follows: in the next chapter we provide a review of the prior works. In the following Chapter 3 and Chapter 4, we discuss two algorithms which takes some assumptions in solving the problem of Knapsack Auctions and we also compute the time complexity.Then in chapter 5, we construct the Knapsack graph and give a new mechanism which solves the problem of overpayment,by giving up on the most optimal solution. It solves the exact Knapsack Problem with near-optimized payment mechanism. Then in chapter 6, we conclude our work.

1.2 Mechanism Design

To implement a solution to an optimization problem where agents have some private information about their preferences for various outcomes, there is a need to optimize the system wide goal. Given agent types, social choice function selects the optimal outcome and it defines the system-wide goal in mechanism design. In order to implement a desirable system wide optimal solution in an equilibrium condition of the induced game, we need mechanism design.

Social Choice Function: The social choice function $f : \theta_1 \times \theta_2 \times \theta_3 \times .... \times \theta_n \longrightarrow O$ [7] maps the players' types to the Outcome of the game.

In other words we can say that, if we are given the types of the agents, i.e, $\theta = (\theta_1, \theta_2, \theta_3, ..., \theta_n)$, then $f(\theta)$ will denote the outcome of the game.

We use mechanism design to implement the rules of a game. For instance, if a mother wants to divide a piece of cake equally between her two children, she would ask one of her child to break it into two pieces and the other child to choose one piece randomly. This way no one has the incentive to cheat.

So we see that, in-spite of the selfish nature of the agents, we need to implement the solution to the social choice function.

Mechanism: Let S_i be the set of strategies available to player i. We denote a mechanism M by $(S_1, S_2, S_3.....S_n, g(.))$, where $g : S_1 \times S_2 \times .....S_n \longrightarrow O$ [7] is an outcome rule. To summarize, we can say that a mechanism is precisely a definition of the available strategies and a way to choose the outcome of the game based on the strategies of the agents.

Mechanism implementation: We say that a mechanism M impements the social choice function $f(\theta)$, if $\forall(\theta_1, \theta_2, ..., \theta_n) \in \Theta_1 \times \Theta_2 \times .....\Theta_n$, $g(s_1^*(\theta_1), s_2^*(\theta_2), ..., s_n^*(\theta_n)) = f(\theta)$ [7] where $(s_1^*, s_2^*, ..., s_n^*)$ is a strategy profile and it is an equiibrium solution to the induced game by the mechanism M.

1.3 VCG Mechanisms

Goal: To design a payment(incentive) scheme in the presence of incomplete information, such that a socially optimal outcome is obtained in dominant strategies.

Intuition: To penalize the players according to the damage they do to the society. Each player pays a "social cost", which is the harm their presence causes to the rest.

Mechanism: The cost to be paid by player i is given by -

$p_i = W'_{-i} - W^*_{others}$

where W'_{-i} is the optimal welfare if player i was not present, and

W^*_{others} is the welfare of the players other than i when the outcome is the one that maximizes the social welfare when all the players are involved.

Formalization:

N: the set of all players

n: the number of players

$w_1, w_2, ..., w_m$: the possible outcomes

$\theta_1, \theta_2, ..., \theta_n$: the private values of each player

$v_i(w, \theta_i)$: value for player i when the outcome is w

Goal of the social planner is to choose the outcome w^* that maximizes the total value for all the players ($\sum_{i=1}^{n} v_i(w^*, \theta_i)$)

To implement the mechanism each of the players is asked to report their private values (θ_i's)

Value reported by the player i is denoted by θ'_i

The outcome that maximizes the total value if i was not playing is denoted by w^*_{-i}

Payment made by each player i is given by:

$p_i(\theta) = \sum_{j \neq i} v_j(w^*_{-i}, \theta_j) - \sum_{j \neq i} v_j(w^*, \theta_j)$

Total utility of player i when the outcome is w^* is given by:

$u_i(w^*, \theta) = v_i(w^*, \theta_i) - p_i(\theta)$

The above expression is equivalent to:

$u_i(w^*, \theta) = \sum_{j \in N} v_j(w^*, \theta_j) - \sum_{j \neq i} v_j(w^*_{-i}, \theta_j)$

The first term in the above equation is the social welfare (W^*) when the outcome is w^*.

The player i can lie and change the outcome.

For any outcome x, utility of player i is given by:

$u_i(x, \theta) = \sum_{j \in N} v_j(x, \theta_j) - \sum_{j \neq i} v_j(w^*_{-i}, \theta_j)$

Since the outcome w^* maximizes the social welfare, for any outcome x,

$\sum_{j \in N} v_j(x, \theta_j) \leq W^*$

Also, the second term is independent of player i. Hence, there is no incentive for player i to lie, since clearly:

$u_i(w^*, \theta) \geq u_i(x, \theta)$

So revealing the true values is the dominant strategy and the social welfare obtained in the dominant strategy equilibrium is optimal.

1.4 Introduction to Knapsack Auctions

Let us present the Knapsack Auctioning problem [1] with the help of an example. Lets say there is TV commercial break agency that wants to sell time-slot to different bidders. The total length of time-slot is W.

There are N bidders, who are advertisers and they want to air their commercial. Each agent i has a commercial of length w_i. with vi being the maximum cost he is willing to pay for airing the commercial.

Let their bid values be $b_1 ... b_n$ and their valuation be $v_1 .. v_n$.

The subset of bidders of total size W denotes the set of alternatives, A. For every $a \in A$ we denote $a_i = 1$ if $i \in a$.

We need to find the optimal alternatives such that $\sum_{i \in A} b_i$ is maximum.

We will try to design a mechanism for payment in Knapsack Auction which follows the following three conditions:

- It should be truthful.
- It should maximize the social welfare function.
- It should be calculated in polynomial time.

Chapter 2

Review of Prior Works

2.1 Frugality

Although VCG mechanisms [7] maximize social welfare, but VCG payments leads to over-payments. We can show that the worst case payment can be less than VCG payments.

Claim: There exists some graph for which the VCG payment is $\theta(n)$ times the second best payment.

Proof: For some arbitrary graph constructed with two paths between source and destination such that path A has n-1 edges having 0 cost and path B has 1 edge of cost 1. The second shortest path has cost 1. According to VCG, the payments will be (n-1)*1 which is $\theta(n)$ times the second best payment.

So now we will try to find payment which we have to make in the worst case by any incentive compatible payment Mechanism. The optimal mechanism, pays logarithmic times the second best cost whereas it can be shown that in the VCG overpays at-least $\sqrt{n}$ times the second best cost. We will focus on mechanisms which maximize revenue, that is, minimize payments.

In this paper we will try to focus on a mechanism that can give a reasonably good solution to the Knapsack Auction problem, but does not make very big over-payments.

2.2 Fractional Knapsack

In 0-1 Knapsack Problem, we cannot break an object. But in Fractional Knapsack we can consider an object partially. To find the new bid value of the object, we find the ration of the fractional weight and use it to calculate the new bid value. To solve this problem, we can use a greedy solution, where we first calculate the ratio of $value_i/weight_i$ for every item. Then we sort these ratios and keep on adding them until the sum reaches W. The last weight may/may not add up-to W, so we can use that weight and its corresponding bid fractionally. The pricing rule for Knapsack Auctions is valid only if:

1. Each object's valuation, which we have selected to be in the knapsack, is greater than or equal to the bid set by agents with similar weights.

2. The summation of the sizes of the weights of the bidders which are chosen, should be at most the capacity of the Knapsack, ie, W.

Chapter 3

Algorithm I

We try to model the Knapsack problem as a Non- Cooperative Game among a set S of n weights with unknown costs, where the weights try to maximise their payoffs. For this we define the value acquired by each weight as the following:

$v_i = x_i * c_i$

where x_i is 1 iff weight i is chosen and c_i is the cost of the ith weight

Now we apply the Vickery Clarke Mechanism described in the previous sections on this problem.

The payment made to the i_th player is

$t_i = W^*_{others} - W^{'}_{-i}$

where $W^{'}_{-i}$ is the optimal welfare if player i was not present, and

W^*_{others} is the welfare of the players other than i when the outcome is the one that maximizes the social welfare when all the players are involved.

Applying it to our problem:

$t_i = (j! = i) \sum_{j=1}^{n} v_j - W^{'}_{-i}$

Since W'_{-i} does not depend on the values told by the ith weight, the ith weight will try to maximise the value of $t_i + v_i$ by maximizing $\sum_{i=1}^{n} v_i$ which is the exact sum, we wish to maximise. This ensures that it is in the best interest of each weight to tell the correct value of their weight and cost.

Since we are following the VCG Mechanism, the mechanism is bound to give us the optimal result.

3.1 Time Complexity Analysis

We analyse the time taken to compute the solution for n bidders be T(n).
We solve the Knapsack Problem using the dynamic programming solution:
knapSack(W, wt, val, n) = max(val[n-1] + knapSack(W-wt[n-1], wt, val, n-1), knapSack(W, wt, val, n-1))
This is a DP of size n*W which would require O(n*W) time to calculate the solution. But W is exponential in the input size. Therefore, as expected, the algorithm takes exponential time.

3.2 Conclusion

We applied the VCG Mechanism on the Knapsack Problem. Since VCG Mechanism is incentive compatible, we are assured of a incentive compatible payment mechanism. But it has it's downsides. The time required to calculate the payment for the n players is exponential in the number of players. Therefore, this method is not a viable solution for real world scenarios. To counter this problem, we use a different approach in the next Chapter.

Chapter 4

2-Approximate Solution

The algorithm presented in previous chapter has exponential time complexity. We further propose an algorithm in this chapter which solves the problem of Payment Mechanism in Knapsack Auctions in Polynomial time. There is a set of n bidders N = {1, . . . , n}.
Let their bid values be b_1...b_n and their valuation be v_1..v_n.
Let the size of each bidder i ∈ {1, . . . , n} be w_i and it is public information.
Let the maximum publicly known weight be W.
The subset of bidders of total size W denotes the set of alternatives, A. For every a ∈ A we denote $a_i = 1$ if i ∈ a.
We need to find the optimal alternatives such that $\sum_{i \in A} b_i$ is maximum.

This problem is NP-Hard, that is if $P \neq NP$, then there is no polynomial time solution for this problem. We need to design a mechanism by which the agents have an incentive to reveal their private valuations truthfully and it should solve it in reasonable amount of time.

4.1 Time Complexity Analysis

The Knapsack problem is weakly NP Hard problem because there exists a dynamic programming algorithm which runs in polynomial time in the problem's dimensions. and the data's magnitude which is involved and is given in integers rather than base-2 log of the magnitude. Hence knapsack algorithm is an exponential function of the size of the input and it is not polynomial.

The Dynamic Programming solution for Knapsack is as follows:

$A[\, i, j\,] = 0 \qquad \forall\, j \in \{0, 1, ..., W\}$

For $i \in \{1, 2, .., n\}$

$\forall\, j \in \{0, 1, ..., W\}$

$OPT[i, j] = maximum(OPT(i-1, j), OPT(i-1, j-w_i) + b_i)$

The running time complexity of the above Dynamic Programming problem is O(n* W) because for each of the n bidders, w_i varies between 1 and W.

So at most there are n * W sub-problems and each sub-problem is solved only once. Hence the running time of the algorithm is O(n*W).

where, $W = 2^{log_2^W}$

and Size of input is log_2^W.

So time complexity is$O(n * 2^{Inputlength})$ which is exponential.

4.2 Algorithm

Let $OPT(b, w) = \sum_{i \in A} b_i$. We will try to solve the approximate Knapsack problem.

Claim: If we find $\sum_{i \in A} b_i \geq OPT(b, w)/2$,then this algorithm is a 2-approximation algorithm for the Knapsack problem.

Proof: If we consider the fractional Knapsack Auction problem, there can be some bidder who can have some weight partially. Let $OPT_F(b, w)$ be the solution of the Fractional Knapsack problem. Therefore, we can write

$$\sum_{i \in A} b_i + b_{i+1} \geq OPT_F(b, w) \geq OPT(b, w)$$

so, maximum $(\sum_{i \in A} b_i, b_{i+1}) \geq OPT(b, w)/2$

We will first sort the bidders in the order of decreasing b_i/w_i ratio.

Now we will rename the bidders as $b_1/w_1 \geq b_2/w_2 \geq b_3/w_3 \geq b_4/w_4 ... \geq b_n/w_n$

Now pick a bidder in a greedy way from starting from b_1. And continue picking up bidders until $\sum_{i \in a} w_i \leq W$.

if $\sum_{j \in a} b_j \geq b_{j+1}$, then return a as the set of alternatives else return b_{j+1}

4.3 Conclusion

In this chapter, we proposed an algorithm which gives an approximate solution to the Knapsack Auctions in Polynomial time. We also proposed a payment mechanism which is incentive compatible as it uses the concepts of VCG Mechanism.

Chapter 5

Frugal Solution

5.1 Frugality ratio

Inspecting frugality of a mechanism is very important while designing the mechanisms for the problems like "hiring a team of people who can get the work done with as less cost as possible",in which case the auctioneer wants to maximize the overall benefit that he gets unlike the case where we assume that there is a central authority who is ready to pay as much as possible for extracting the true cost from each of the agents thus aiming for social welfare.[3]

Intially Frugality ratio was defined as payment that is made to the feasible winning set **S**,divided by the cost of the cheapest possible set **S**' which is disjoint from the set **S**. In this way the overpayment of the mechanism was measured. But this analysis is not useful when there is no other feasible set **S'** disjoint from the set **S**, for example consider the graph which has a single cycle and all the vertices are part of this cycle, i.e., there are no edges outside this cycle, in this graph there is no monopoly and also we can see that there is no other set **S'** which is disjoint from **S**. As a result the frugality ratio of a mechanism was redefined.

We give a formal definition of "hiring a team auctions that we study. A set system (**E,F**)

is defined by a set **E** of n agents, and a collection F $\subseteq 2^E$ of feasible sets.[5] Set system is said to be monopoly free if there does not exist any element that is there in all the feasible sets. There is also a private information associated with every agent e which is the cost c_e, it is the true cost that is incurred if this agent is present in the final solution.

To capture the overpayment of a mechanism, w.r.t a natural lower bound, we define a new notion of the frugality ratio.Since we are interested in a natural lower bound, we wish to define the cheapest Nash value $v(c)$ to be the minimum payments by the mechanism over all of its Nash Equilibria but unfortunately Nash Equilibrium does not exist for some mechanisms but from the intuition gained from Nash Equilibrium, we can still define the quantity $v(c)$.

Definition: In a set system (**I, H**) with **S** as the cheapest feasible set w.r.t the true costs c_e, (c) is the solution to the following optimization problem,
Minimize $\sum_{e\in S} b_e$ subject to,
1) $b_e \geq c_e$ for all e
2) $\sum_{e\in S\setminus T} b_e \leq \sum_{e\in T\setminus S} c_e$ for all T in H
3) For all $e \in S$, we have $T_e \in H$, e $\notin$T$_e$ such that $\sum_{e'\in S\setminus T_e} b_{e'} = \sum_{e\in T\setminus S} c_e$ [5]

The first condition states that none of the agent is ready to bear the loss. Second condition states that out of all feasible sets,**S** has the lowest possible total bid and the third condition explains why none of the agents in **S** can not increase their bids and still be able to increase the profit.

Consider the case of u-v path auction in which there are some number of feasible paths. Then, (c) is going to be the cost of the second-cheapest path. Our intuition is that the agents belonging to the lowest cost feasible path will raise their bids until the sum of their bids equals the cost of the second lowest feasible paths, after which they can not further raise their bids. None of the other edges belonging to non-optimal feasible solution will

have an incentive to change their bids because increasing the bids will make them lose anyway and decreasing the bids will bring them loss.

Definition: For a set system (E, F) and a mechanism M and actual costs $\mathbf{c}$, if the total payments made by mechanism is $P_M(C)$,then the frugality ration of the mechanism M is given by as in [5],

$$\phi_M = \sup_c \frac{P_M(C)}{v(c)}$$

5.2 r-out-of-k Systems

In this section, we discuss a competitive mechanism with near-optimal frugality ratio for the r-out-of-k set systems.[5]

Definition: A set system $(E, \mathcal{F})$ is an r-out-of-k system if there exists a partition of E into k disjoint sets S_1,S_2...,S_k, such that every set $F \in \mathcal{F}$ contains exactly r out of these k sets of elements.

This system generalizes the problem of finding a spanning tree on a cycle (with k sets S_i's corresponding to each edge, and r = k-1). It also generalizes the problem of finding the shortest path among the k vertex-disjoint paths (with k S_i's corresponding to the k paths, and r = 1).

Intuition: It can be observed that for a set chosen as winner, each element has to be paid a certain amount (which depends on the bids of the other sets). The largeness of the selected set tends to increase the payment. Hence a mechanism should give (reasonable) preference to smaller sets. The r-out-of-k system mechanism achieves this by comparing the costs c(S_i) weighted by the factor γ_i. γ_i represents the relative size of S_i compared to the other sets. For a (k-1)-out-of-k-system, γ_i is defined as the solution of the system of equations given by:

$$\alpha = \frac{1}{(k-1)\gamma_i} \sum_{j \neq i} \gamma_i |S_j| \quad for\ i = 1, 2, ..., k \tag{5.1}$$

The proof that there exists a solution to the system of k equations above, is provided in [5]

.

For a general r-out-of-k system, the most expensive (k-r-1) sets are discarded, and the problem for r-out-of-(r+1) system is considered. The mechanism [5] is as follows:

Input: An r-out-of-k system with bids b_e for each element in E.

1. Rename the sets S_i's so as to have them sorted by non-decreasing sums of bids b(S_i).

2. Keep the r+1 cheapest sets, and discard the rest.

3. For $1 \leq i \leq r+1$, let γ_i be as defined by the equation (1).

4. Let $l = argmax_i \ \frac{b(S_i)}{\gamma_i}$.

Output: F = $\cup_{i \neq l} S_i$

Theorem: For any value of r in 1 and k-1, the above mechanism is truthful and competitive (the frugality ratio is within a constant factor of the optimal).

Lemma: For a (k-1)-out-of-k system (E,$\mathcal{F}$), α defined by equation (1), the frugality ratio of any truthful mechanism for this system has a lower bound of $\alpha/2$.

The proof of the above stated Theorem and Lemma can be found in [5].

5.2.1 1-out-of-2-systems

Consider the problem of selecting one out of the two disjoint sets S_1 and S_2. Using the lemma, any truthful mechanism must have a frugality ratio of $\frac{1}{2}\sqrt{|S_1||S_2|}$ [5]

Lemma: Any truthful mechanism $\mathcal{M}$, for a 1-out-of-2 system with sets S_1 and S_2 has a frugality ratio which is at least $\frac{1}{\sqrt{2}}\sqrt{|S_1||S_2|}$

5.3 The mechanism:

In this paper, we try to minimize the payments in the 0-1 Knapsack Auction Problem. We will first start by showing that knapsack problem can be reduced to costliest path problem on a directed acyclic network.[4] Then we will devise a mechanism which does not pays an optimal payment to the agents in order to keep their valuations truthful.

We first give an example of the type of problem that we are looking to solve. Lets say there is TV commercial break agency that wants to sell time-slot to different bidders. The total length of time-slot is W. There are N bidders, who are advertisers and they want to air their commercial. Each agent has a commercial of some length along with a valuation being the maximum cost he is willing to pay for airing the commercial.

Maximise $\sum_{i=1}^{n} b_i x_i$

Subject to $\sum_{i=1}^{n} w_i x_i \leq W$

There is a set of n bidders N = {1, . . . , n}. Let their bid values be $b_1, b_2, ..., b_n$ and their valuation be $v_1, v_2, ..., v_n$ Let the size of each bidder i $\in$ {1, . . . , n} be w_i and it is public information. Let the maximum publicly known weight be W. The subset of bidders of total size W denotes the set of alternatives, A. For every a $\in$ A we denote a_i = 1 if i $\in$ a.

We need to find the optimal alternatives such that $\sum_{i \in A} b_i$ is maximum.

5.3.1 Construction of Knapsack Graph

We first formulate this problem as a path problem in the following way:

Let $G = (V, E)$ be a graph with a set of nodes V=$\{0, 1, ..., W\}$. The node with label 0 corresponds to the source vertex and the node with label W corresponds to the destination vertex. A node with label v implies that on reaching this node, we have filled our knapsack with total weight v.

The set of edges E is composed of two types of directed edges. We have edges of the form $(w, w + w_i)$ such that $w + w_i \leq W$, with bid b_i.
We also construct $|W|$ edges of the form (y, W) each with a bid 0, where $y \in \{0, 1, ..., W-1\}$. This is explained in detail in the following section.

Construct the nodes $V = \{0, 1..., W\}$ with no edges.;

Mark the nodes 1....,W to be unreachable.;

Mark node with label 0 as reachable and construct an edge (0,W) with cost=0;

set v to be 0 ;

while $v < W$ **do**

  if *w is reachable* **then**

    set i to be 0;

    while $i < n$ **do**

      if $v + w_i \leq W$ **then**

        if $v + w_i$ *is not reachable* **then**

          mark $v + w_i$ to be reachable;

          construct an edge ($v + w_i$, W) with bid=0 and add it to the set E;

        end

        construct an edge (v, $v + w_i$) with bid= b_i and add it to the set E ;

      else

        increment i by 1;

      end

    end

  else

    increment v by 1;

  end

end

Algorithm 1: Constructing the Knapsack Graph G=(V,E)

5.3.2 Selection rule

Till now, we haven't taken into account the number of agents involved while choosing the optimal solution. If the winner is a set with large number of agents, then each of the agent will have to be paid a certain amount. This increases the overall payment. Using the Knapsack Graph, we take into account the cardinality of the set and the total valuation of the set, which will help us achieve near-optimal payment mechanism.

We extend the r-out-of-k-system[5] mechanism to the 1-out-of-2 system[5] mechanism for the Knapsack Graph Problem. We now consider this directed acyclic Knapsack graph G. We now select two agent disjoint paths P_1 and P_2 such that $\Sigma_{i \in P_1} b(i) + \Sigma_{i \in P_2} b(i)$ is maximum.

Let $V_{common} = v_1, v_2, v_3, ..., v_{k+1}$ be the vertices that P_1, P_2 have in common, in the order in which they appear in P_1 and P_2. Here v_1=vertex 0 and v_{k+1}=vertex W.

We denote the path from vertex v_i to v_{i+1} by P_1^i for Path 1 and P_2^i for Path 2.

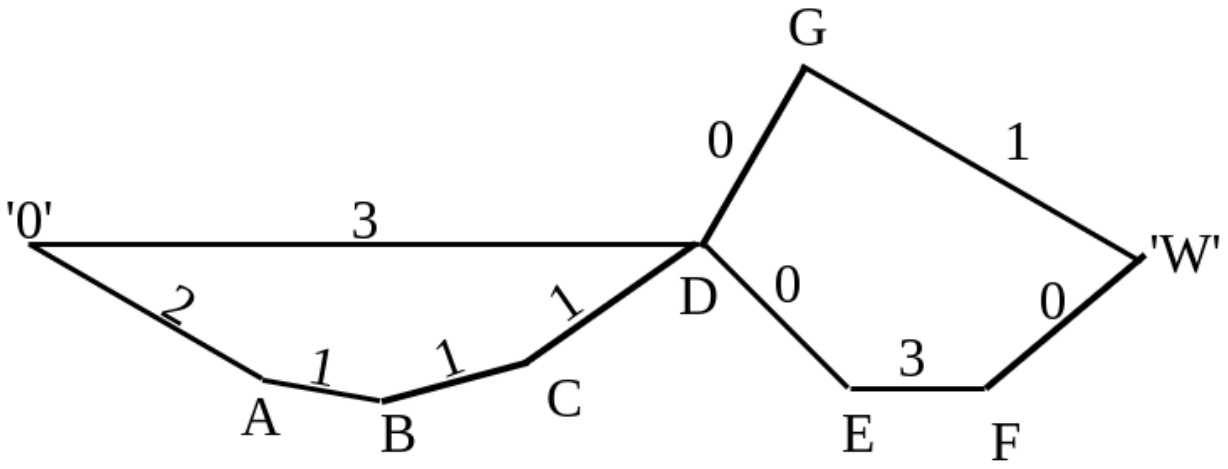
For all i between 1 to k, we will include P_1^i in the solution,if

$$\frac{\Sigma_{i \in P_1^i} b(i)}{\sqrt{\|P_1^i\|}} \geq \frac{\Sigma_{i \in P_2^i} b(i)}{\sqrt{\|P_2^i\|}}$$

, else we will include P_2^i in the solution.

So we get a Path from 0 to W which pays optimally to the agents which we will describe in the following section. The payment to each agent would be, the maximum he could bid such that, if bid of other agents is kept same, he would still win.

We will explain this with an example.



Let the two costliest paths be $P_1 = (0, D, G, W)$ and $P_2 = (0, A, B, C, D, E, F, W)$. Here $v_1 =' 0', v_2 =' D', v_3 =' W'$.

For, the path between v_1 to v_2, we have $P_1^1 = (0, D)$ and $P_2^1 = (0, A, B, C.D)$

The payments to the winning agents are as following:

Since $\frac{3}{\sqrt{1}} \geq \frac{5}{\sqrt{4}}$

So Agent(0,D) pays $\frac{5}{\sqrt{4}}$

For, the path between v_2 to v_3, we have $P_1^2 = (D, G, W)$ and $P_2^2 = (D, E, F, W)$ Since $\frac{3}{\sqrt{3}} \geq \frac{1}{\sqrt{1}}$

So Agent(D,E) pays $\sqrt{3} - 3$, Agent(D,E) pays $\sqrt{3}$, Agent(F,W) pays $\sqrt{3} - 3$

5.3.3 Proof of Truthfulness of our Mechanism

We now prove that our mechanism is **Monotone**, i.e., for a mechanism to be truthful, the selection rule must be monotone. So we need to prove that none of the losing agents can be a winner by lowering their bids, such that other agents' bids aren't changed.

Case (i): If the agent is not a part of the paths P_1 and P_2 such that cost$(P_1 + P_2)$ is maximum, then if he lowers his bid, the path containing him, still won't be selected as $P1$ or $P2$.

Case (ii) If the agent is part of P_1 or P_2 but our mechanism does not select him, then if we lowers his bid, he still won't be selected. This can be shown mathematically as following. Let the agent not selected in the winning set be agent 'k'.Without loss of generality, let agent 'k' be part of P_1 between v_i and v_{i+1}, i.e., agent 'k' lies on the path P_1^i. Since it is not part of the winning set, it implies $\frac{\Sigma_{i \in P_1^i} b(i)}{\sqrt{\|P_1^i\|}} < \frac{\Sigma_{i \in P_2^i} b(i)}{\sqrt{\|P_2^i\|}}$. So if agent 'k', lowers his bid, still P_2^i will be selected.

Hence, we have proved that our mechanism is monotone. Hence it is a truthful mechanism.

Chapter 6

Conclusion

We have discussed the basic principles of Mechanism Design , VickreyClarkeGroves mechanism and Knapsack Auctions. We tried to devise a payment mechanism for Knapsack Auction problem, but saw that solving he exact Knapsack problem has an exponential time complexity. So we approximated the Knapsack Problem and gave a solution which is at-least half as optimal as the solution to the Exact Knapsack Problem. Using this solution, we devised a payment mechanism, by which we could insure that the agents have no incentive to lie about their bidding value.

In our last chapter, we devised a frugal mechanism for Knapsack Auctions, which minimizes the payments. Hence the problem of overpayment associated with the Knapsack Auctions is solved. This mechanism takes into account both the cost acquired from the agents and the number of agents in the selected set. Hence, by giving up on the most optimal solution, we can minimize our payments which are made to the agents to keep the mechanism incentive compatible.

References

[1] G. Aggarwal and J. D. Hartline. Knapsack auctions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1083–1092. Society for Industrial and Applied Mathematics, 2006.

[2] A. Archer and É. Tardos. Frugal path mechanisms. *ACM Transactions on Algorithms (TALG)*, 3(1):3, 2007.

[3] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 701–709. Society for Industrial and Applied Mathematics, 2004.

[4] A. M. Frieze. Shortest path algorithms for knapsack type problems. *Mathematical Programming*, 11(1):150–157, 1976.

[5] A. R. Karlin and D. Kempe. Beyond vcg: Frugality of truthful mechanisms. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 615–624. IEEE, 2005.

[6] V. Krishna. *Auction theory*. Academic press, 2009.

[7] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.