Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 04

Lecture 18

Lecture 18 : FPTAS for Knapsack

Welcome. So, in the last class we have seen a dynamic programming based algorithm which runs in pseudo polynomial time for the knapsack problem. So, today we will use that algorithm to design a fully polynomial time approximation scheme also known as FPTAS for the knapsack problem. So, today's topic is an FPTAS. for knapsack. So what was the running time of our dynamic programming based algorithm for knapsack? dynamic programming based algorithm for knapsack runs in time b go of n times minimum of b and v, where n is the number of items B is the capacity of knapsack and V is sum of values of the item.

The idea for designing FPTAS for knapsack is to scale down the sizes of the items, so that and b also. So, that we can use this algorithm which runs in polynomial time if one of b or v is polynomial if one of B and or V is polynomial related polynomial upper bounded of n. So, minimum of B and V is less than equal to poly n. So, the idea is to bring down minimum of B, V to be bounded above by some polynomial function of n without compromising the quality of the solution too we have to compromise the quality of the solution to some extent because knapsack is a NP complete problem and hence assuming $P \neq NP$ we do not hope to have a polynomial time algorithm for knapsack.

But we want to minimize the compromise we will make the quality of the solution. So, for that we scale down the values of every item by some scaling factor say μ whose value we will decide later we will find out the value of mu which we need to set. So, that our compromise to the quality of the solution is small. So, formally we define $v_i' = \lfloor \frac{v_i}{\mu} \rfloor$ and the sizes remain same the sizes of the items remain the same. By bringing down the value of every item what we have essentially obtained is let us define $V' = \sum v_i'$ which is $\sum \lfloor \frac{v_i}{\mu} \rfloor$ which is less than equal to

$\frac{1}{\mu}\sum v_i$ which is $\frac{V}{\mu}$. So, let us call the new instance I' and observe that set of items. is feasible in I that is their sum of sizes is less than equal to the capacity of the bag if and only if it is feasible in I' ok. Now, let us find out the value of μ for that the dynamic programming based algorithm for knapsack runs in time $O(nV')$ for the instance I'. Why this is the case? Because it turns in time we go of n times minimum of V' and B and we have we will choose μ in such a way that $\frac{V}{\mu}$ will be less than B. So, towards that let m be the maximum value of any item in the instance I that is $M = max_{i=1}^{n} v_i$, then we observe that $V \leq nM$ ok. also because we have assumed without loss of generality that size of every item is less than equal to the size of capacity of knapsack then M is a lower bound on opt. Moreover, since we have assumed without loss of generality that $s_i \leq B$ for all $i \in [n]$, it follows that opt which is the optimum value of a set of items whose total size is less than equal to B, $opt \geq M$. So, you see using M we get a very good handle on opt that is we have opt is in between v which is nM.

So, let us write nM and greater than equal to m ok good. Now, it makes sense thus it makes sense to set μ in such a way that $n\mu$ what could be the maximum difference of values. for any set S of items V of S which is the sum of the values of the items in S. This how this thing compare with $\mu v'(S)$. So, this is $v'(S)$ this is $\mu \times \sum v_i'$ which is $\mu \times \sum \lfloor \frac{v_i}{\mu} \rfloor$ which is of course, less than equal to $\mu v'(S)$ this is less than equal to v(S), but this is, but how much it can it can be less.

So, for that $\mu v'(S)$ which is $\mu \times \sum \lfloor \frac{v_i}{\mu} \rfloor$. Now, each of this quantity differs can differ from v_i by at most μ. So, this is greater than equal to $\mu \times n$ since $\frac{v_i}{\mu}$ if you divide any integer with some number and you take the floor and you multiply this number, then this compares it with v_i this can drop by at most μ.

So, this is $v(S) - n\mu$ ok. Now, we want our alg. So, what is the algorithm? The algorithm is you take the items scale down their values solve that and output that that set. So, let us write the pseudo code of the algorithm. that will help us to determine what is the value of μ.

So, we will set v_i' to be $\frac{v_i}{\mu}$. So, we will set mu we will see what value of μ in we need to set this is for all $i \in [n]$. and then we use the dynamic programming algorithm to obtain an optimal solution say S which is a subset of items. of I', what is I'? The same set of items remain their values becomes just v_i' and their sizes remain same ok and then you output S. So, this is the pseudocode.

So, what is ALG? ALG is an optimal solution of I'. Now, ALG equal to $v(S)$ where S is an optimal solution of I' this is exactly what we are doing we are scaling down the values getting an instance I' we are solving it getting an optimal solution S and simply outputting that set of items. So, this is $v(S)$. I want to show that this is not too less. So, Now, I need to use that S is an optimal solution of I' that is why I need to connect $v(S)$ with $v'(S)$. So, $v(S)$ I want to write greater than equal to because I want to show ALG is greater than equal to something.

Now, here I will use the inequalities. There are two inequalities which connect $v(S)$ with $v'(S)$. One is one is here what we have is from here that $v'(S)$ is less than equal to $v(S)$ this is one inequality and the other one is here that $v'(S)$ is greater than equal to $\frac{v(S)}{\mu}-n$. So, these are the two inequalities we have. $v(S)\geq\mu v'(S)$ because in v' we are dividing by μ in v we are not dividing by μ it original valuations remain. So, you have let us write it clearly μ times sorry we have $v(S)$ is greater than equal to $\mu v'(S)$. So, let us write there $v(S)\geq\mu v'(S)$, and now $\mu v'(S)$ is greater than equal to $v(S)-n\mu$ ok. I want this to be this to be small with respect to opt. So, we want $n\mu$ to be at most $\epsilon\times opt$, but we do not know opt, but we know opt is at least M. So, it is enough. to ensure that $n\mu$ is at most epsilon times M which in turn ensures that $n\mu$ is at most $\epsilon\times opt$.

So, one way to guarantee that is to set $n\mu$ equal to $\epsilon\times M$ which gives us what μ to set μ equal to $\epsilon\frac{M}{n}$. So, we set this μ ok. Now, we finish the algorithm. So, this is $\epsilon\frac{M}{n}$ where M equal to maximum valuations of any item ok. So, with this now let us finish the analysis that it is a $1+\epsilon$ factor approximation algorithm.

So, let S be the output of the algorithm as usual. sorry let us start with ALG, ALG equal to $v(S)$ which is summation which is we need to go to I' see which is greater than equal to $\mu\times v'(S)$. Now, S is an optimal solution of I'. So, S is an optimal solution of I' and I want to connect it to opt. So, for that let O be an optimal solution of I.

So, for I' I know S is an optimal solution. So, this is greater than equal to $\mu\times v'(O)$ this is so, because S is an optimal solution of I'. Now, I use this fact for any set S I know that $\mu\times v'(S)$ is greater than equal to $v(S)-n\mu$ here this holds for all set $S\subseteq[n]$. So, this we use here this is greater than equal to $v(O)-n\mu$ ok. $v(O)$ is nothing, but $opt-n\mu$ is nothing but ϵM.

Now, ϵM is a lower bound on opt that means, M is at least opt this is greater than equal to $opt - \epsilon \times opt$ since opt is greater than equal to M which is $(1-\epsilon) \times opt$. Hence, it outputs a set which is set of items whose total size is less than equal to capacity of the bag and its value is at least $1-\epsilon$ times the optimal function optimal set and the runtime of the algorithm. runtime see it is $O(n \times V')$. Now, recall what is the runtime of the original dynamic programming algorithm, it is n times minimum of V' and B and minimum of V' and B is less than equal to V'.

So, this is $O(n \times V')$ V' of any set is less than equal to $v(S) - \mu$. So, this is $\frac{v}{\mu}$ and what is μ? μ is $\epsilon \frac{M}{n}$. $O(\frac{n^2 \times V}{\epsilon M}) = O(\frac{n^3}{\epsilon})$.

So, you see the runtime is polynomial in n and ϵ. So, let us stop here. Thank you.