Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 05

Lecture 22

Lecture 22 : An APTAS for Bin Packing

Welcome, in the last lecture we have seen a polynomial time approximation scheme for scheduling jobs on multiple identical machines. So, today we will discuss another very important problem in computer science which is called bin packing and we have seen an approximation algorithm for bin packing. So, today's topic is bin packing. So, what is the input? Input are n items with sizes $a_1, a_2, \ldots, a_n$ where each a_i is greater than 0 and less than 1 and the goal is to partition these n items into minimum number of parts. So, that the sum of the sizes of the items in any part is at most 1. So, you can think of these items are sizes and there are bins or which with capacity 1 and I want to use minimum number of bins to pack all these items.

So, this problem is known to be NP complete even for checking whether it is possible to pack all the items into 2 bins theorem. computing if given set of items can be packed. into 2 bins is NP complete. There is a very easy reduction from the partition problem and you can take it as a homework.

reduction from the partition problem, details are homework. So, what is partition problem? Let me tell you partition problem we are given n items again with their sizes. say $b_1, \ldots, b_n$ just want all b_i's greater than 0 for all $i \in [n]$. We do not need all b_i's to be strictly less than 1, they can be more than 1 also and the goal is to check if there exists a subset I of items such that. summation b_i the sum of the sizes of the items in set I is same as the sum of the sizes of the items in set $[n] \setminus I$.

So, this is the partition problem you start with a partition problem any arbitrary instance of it and reduce an equivalent instance of knapsack thereby showing that the knapsack problem even in the special case whether 2 bins suffice or not is NP complete. So, immediate corollary from this theorem is and partition problem is known to be NP An immediate corollary of this theorem is that there does not exist any better than $\frac{3}{2}$ factor approximation algorithm for bin packing theorem. There does not exist any ρ factor approximation algorithm for bin packing. for any $\rho<\frac{3}{2}$. Because we have seen this sort of proof again in this course that if there exist a better than $\frac{3}{2}$ factor approximation algorithm for bin packing, then we can use this algorithm to solve this instance, we can use that algorithm to compute if a given instance of bin packing can be can if the if the if a given set of items in a bin packing instance can be packed in 2 bins or not.

but in bin packing we will show something remarkable which is called asymptotic PTAS. So, let us first define it what is APTAS or asymptotic asymptotic polynomial time approximation scheme APTAS in short. What is it? An APTAS is a family of algorithms ϵ parameterized by ϵ for every ϵ greater than 0 like PTAS or FPTAS and a constant c such that the algorithm A_ϵ returns and returns are $(1+\epsilon)\times opt+c$. approximate solution or returns a solution of value at most $(1+\epsilon)\times opt+c$ this is for minimization problems. So, what we will see that for knapsack there exist an EPTAS.

So, this is the theorem that we will prove now and this runs in time in time the running time is like PTAS $n^{O(f(\epsilon))}$. So, for every constant epsilon this is a polynomial time algorithm ok. So, for every epsilon greater than 0 there exists an algorithm for knapsack for bin packing. which packs the items into at most $(1+\epsilon)\times opt+1$ bins in time $n^{O(f(\epsilon))}$. So, we will see what is the exact running And the idea is we will use the our algorithm the PTAS for the job scheduling and the high level idea is also same that we will divide the items into small item and large item and focus only on large items.

So, for a parameter γ whose value we will decide at the end of the algorithm. We call item i large $i\in[n]$ if $a_i\geq\gamma$. else we call a_i we call the item i small. So, here is an important lemma that any packing of large items into l bins can be extended. into extended to packing of all items into $max\left\{l,\frac{1}{1-\gamma}SIZE(I)\right\}$ bins where SIZE(I) is the sum of the sizes of all the items in the instance ok.

proof. So, we start with any packing of large items into l bins and then we try to greedily pack all the items all the remaining items which are small into the into the bins. What do you mean by greedily pack? We pick small item in every say item j in every iteration of our greedy algorithm. If there is a bin with free space a_j, then we put item j on that bin. Otherwise we put j into a new bin. So, that is the case.

So, two things can happen, if we start this our greedy algorithm from the packing of large items into l bins and every time I pick a small item, if it is the case that we do not need to open a new bin and pack all small items, then in that case we are able to pack all items into l bins. The other case when at least one new bin is opened, there we will show that the number of bins used is at most $\frac{1}{1-\gamma} \times SIZE(I)$. So, if the greedy algorithm does not open any new bin, then we use we pack all items into l bins. On the other hand, if a new bin is opened then j be or then let k be the iteration when the last new bin is opened. Let j be the small job picked in iteration k.

then its size is a_j the size of job j which is less than gamma and that item is not fit in is not fit in any of the existing bins. That means, that all the existing bins must be at least $1-\gamma$ full actually more than $1-\gamma$ full of the bins, none of the existing bins have free space at least γ. Then we observe that none of the existing bins has free space at least γ ok. Let alg be the number of bins used by the algorithm.

Then at least $alg-1$ bins are more than $1-\gamma$ full, then look at $alg-1$ each of these bins except the last bin opened each of the existing bins they are kept the size some of the sizes of the items in this bins each of this bin is at least $1-\gamma$ ok. This should be less than equal to SIZE(I). because SIZE(I) is the sum of the sizes of all the items. From here we get $alg-1$ is less than equal to $\frac{1}{1-\gamma} \times SIZE(I)$, hence alg is less than equal to $\frac{1}{1-\gamma} \times SIZE(I)+1$ ok. And before we have shown that in this case alg is at most l.

So, we have alg is less than equal to $max\left\{l, \frac{1}{1-\gamma} SIZE(I)\right\}+1$. Now, see that SIZE(I) is a lower bound on opt minimum number of bins. So, this is also less than equal to $\frac{1}{1-\gamma} \times opt(I)+1$. So, we will use this crucially. So, there also we have seen that if we are able to pack the large items in optimal or approximately optimal number of bins, then by applying this greedy algorithm from the from the packing of large items we will get an approximately optimal solution.

if we are able to pack large items optimally. or approximately optimally, then our greedy algorithm outputs an approximately optimal packing. So, we will see how this idea is implemented in the next lecture. Thank you.