Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 05

Lecture 23

Lecture 23 : An APTAS for Bin Packing Contd.

In the last lecture we have started seeing an EPTAS for bin packing problem and we will finish that idea. So, we have seen that it is enough to pack the large items either optimally or even approximately optimally to pack all items in approximately optimal number of bins. Recall the large items are defined by those items whose size is at least more than gamma and the small items are items which are whose size is less than gamma. So, now, we will see an algorithm to pack the large items in approximately optimal number of bins. First task is approximately optimally pack large items.

So, recall large items are items item i is large if $a_i \geq \gamma$. And the idea for this approximately optimal packing is at a very high level is similar to the dynamic programming algorithm for job scheduling. In the dynamic programming algorithm for job scheduling, we use rounding to bound the number of different processing times of the jobs. Here also we will do something similar, but importantly different also to bound the number of different item sizes. Here the idea is what is called linear grouping.

So, let k be a parameter whose value we will decide at the end of the algorithm depending upon the requirement like gamma, gamma also its value we have not set yet gamma and k this will be set at the end of the algorithm. So, that our algorithm runs in polynomial time and it outputs a solution with size at most $(1+\epsilon)\times opt+1$ number of bins. So, let k be on an integer parameter. So, by renaming the items we can assume that their sizes are monotonically non-increasing . We can assume without loss of generality by renaming the items that $1>a_1 \geq a_2 \geq a_3 \geq \ldots \geq a_n$.

Now, what we do is we partition this items into buckets of size k. So, the first group is a_1 up to a_k a_{k+1} up to a_{2k}, a_{2k+1} up to a_{3k} and so on up to a_n. This is the first bucket, this is size k. this is the second bucket next k items, this is the third bucket next k items and so on. Unless k divides in the last bucket has items less than k in general it has items less than equal to k.

Now, what we do we in the rounded down instance equivalent to the rounded down instance what we do we throw away the top k items a_1 to a_k and replace this a_{k+1} to a_{2k} with the maximum a_{k+1}, then the next item let us call it a'_{k+1} which is same as a_{k+1}, but the next item size a'_{k+2} that is also a_{k+1}, a'_{2k} is also a_{k+1}. That means, in each group of k items we are replacing every item with an item of maximum size in that group. So, this is the after changing the weights these sizes these are the this is how the top group of k items look like. So, in the next group a'_{2k+1} is a_{2k+1}.

here in the next group the maximum size is a_{2k+1}. So, we replace each item with another item of size a_{2k+1}. and so on. This instance I am calling I the original instance for which I want to compute a packing and this instance I am calling I' the modified instance. What is the relationship between I and I'? So, here is an important lemma that $opt(I)$ is less than equal to $opt(I')+k$ and greater than equal to $opt(I')$. So, let us prove this lemma and once we prove this lemma we will put suitable values of k and we will show how we can solve this instance in I' you see the number of different sizes are not too much it is it will be small we will see, but before that we prove this moreover any packing of I' can be extended to obtain a packing of I using at most k additional bins.

easy both are easy. So, first let us show that to show $opt(I')$ is less than equal to $opt(I)$. What you do for that? Consider any packing of I into say l bins. Now, in this packing you replace the i-th item with $(k+i)$-th item. in this packing we replace the i-th item of the instance i with the $(k+i)$th item of I' and we claim that the resulting packing is valid because valid means the sum of sizes of all the items assigned to 1 bin that is less than equal to 1.

The resulting packing is valid this and through the last items of last bag that is not important the resulting packing is valid for I'. packing is valid since now we are replacing the i-th item of instance I whose size is a_i with the $(i+k)$-th item of I' this is for all $i \in [n-k]$. because this is a_i is greater than equal to a'_{i+k} because a_i and a_{i+k} to consecutive bag and a'_{i+k} is the maximum i size of the bag where it belongs and, but that is less than equal to a_i. So, we are replacing each item with a smaller item and the resulting packing valid since it packs all items of I' and maybe something more. So, we have that $opt(I')$ is less than equal to $opt(I)$.

the other direction is also simple. So, we start with a packing of $opt(I')$. Next to show $opt(I)$ is less than equal to $opt(I')+k$. We start by the way for this proof the. indices of

I' starts from a_{i+k} here in the for i the indices start from 1 to n the items in i prime are indexed with $k+1$ to n ok.

We start with any packing of I' and the idea is if you look at a_i' and a_i a_i' can only be bigger for $i \in \{k+1, \ldots, n\}$. So, we replace a_i' with a_i. for all $i \in \{k+1, \ldots, n\}$ ok. This is a valid packing since a_i' is greater than equal to a_i for all $i \in \{k+1, \ldots, n\}$. So, this is a valid packing that is why, but in instance I has k more items which we can pack individually in each we use k new bins and we can put a_1 in the first new bin a_2 in the second new bin and a_k in the k-th new bin.

And hence we have proved that $opt(I)$ is less than equal to $opt(I')+k$ we can pack the k extra items of i into k new bins. Hence, $opt(I)$ is less than equal to $opt(I')+k$. Moreover as you see that we can this part this direction we have started with a packing of I' and we have simply seen that item as the original item and we have extended that packing to a packing of I using at most k extra bins. Now we will say how we get an approximately optimal solution for this problem ok. So, for that we set Now, we set up various values of k and ϵ.

The number of distinct pieces of items in I' is at most is $\left\lceil \frac{n}{k} \right\rceil$. see the number of groups is $\left\lceil \frac{n}{k} \right\rceil$ and we are throwing away the first group. So, the number of groups for the for I' which is same as the number of distinct items is at most floor of $\left\lceil \frac{n}{k} \right\rceil$. which is $\frac{n}{k} - 1$ because we are throwing away the first group which is at most $\frac{n}{k}$ ok. And now we set since So, we set γ to be $\frac{\epsilon}{2}$ recall in last lecture we have discussed that we can if we can pack the large items into l bins and then we can if we can pack large items into l bins, then we can pack all items in $max\left\{l, \frac{1}{1-\gamma} SIZE(I)+1\right\}$ bins.

Now, we discussed that $\frac{1}{1-\gamma} SIZE(I)+1$, $SIZE(I) \leq opt$. So, this is $\frac{1}{1-\gamma} opt(I)+1$. Now we want to set γ. So, that this is less than equal to $(1+\epsilon) \times opt(I)+1$.

So, set γ. Now, you can check that if I set γ to be equal to $\frac{\epsilon}{2}$, then this is enough because

$\frac{1}{1-\frac{\epsilon}{2}}$ is less than equal to $1+\epsilon$, this you can prove using elementary calculus ok good. that is why we put if γ equal to $\frac{\epsilon}{2}$. So, the number of large items in I we do not have any small item n is small items. So, $SIZE(I)$ is at least because we are setting γ to be $\frac{\epsilon}{2}$ at least $\frac{\epsilon}{2}\times n$ because size of each item is at least $\frac{\epsilon}{2}$.

So, if we k to be floor of ϵ of $SIZE(I)$, then we see that $\frac{n}{k}$ which is the number of maximum number of distinct items in the after doing the linear grouping in the round down instance I' is less than equal to $\frac{2n}{\epsilon\times SIZE(I)}$ because k is this. So, this is and $SIZE(I)$ is greater than equal to $\frac{\epsilon}{2}\times n$. So, this is less than equal to $\frac{4}{\epsilon^2}$. So, here we are using the fact that floor of any α is any real number α is greater than $\frac{\alpha}{2}$ when α is greater than equal to 1. So, there are so, we assume that $\epsilon\times SIZE(I)$ is greater than equal to 1 since otherwise the number of items is at most $\frac{\frac{1}{\epsilon}}{\frac{\epsilon}{2}}$ because $SIZE(I)$ is if it is less than 1 then sum of the sizes is at most $\frac{1}{\epsilon}$ and each size of every item is at least $\frac{\epsilon}{2}$.

So, which is $\frac{2}{\epsilon^2}$ and because we can have only this much items we can apply the dynamic programming algorithm. to solve I' optimally. Hence, after linear grouping we are left with constantly many constantly many pieces which we can solve optimally. Hence what we have shown is this theorem that for any ϵ greater than 0 there is a assuming ϵ is constant that packs items into at most $(1+\epsilon)\times opt+1$ bins. So, how do you prove this? Proof.

So, we set k to be less than $\lfloor\epsilon\times SIZE(I)\rfloor$. Now, there are two cases if k is greater than equal to 1, then we can solve this part after the round down instance optimally and then we can use this result to get a packing with $1+opt(I)$ bins otherwise we have. So, this case is done if k is otherwise k is less than equal to 1 and then we again find an optimal packing after the round down instance and get this $(1+\epsilon)\times opt+1$ bins with this bins we pack all the items ok. So, let us stop here. Thank you.