Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 04

Lecture 20

Lecture 20 : PTAS for Minimizing Makespan for Scheduling Jobs on Parallel Identical Machines

Welcome in the last class we have seen a PTAS for minimizing make span of jobs into identical multiple machines, when the number of machines is constant. Now, in this class we will remove that requirement that the number of machines is constant. So, it is again we will see a PTAS for the general case. So, today's topic is PTAS a polynomial time approximation scheme for job scheduling on multiple parallel machines. So, if you recall let us briefly recall the last algorithm. So, we have a lower bound on opt which is greater than equal to $\frac{1}{m}\sum p_i$ and we divided the jobs with among long jobs and short jobs .

So, long jobs are those whose processing time s greater than equal to $\frac{1}{k \times m}\sum p_i$ where k is a parameter of the algorithm A_k which will output $\left(1+\frac{1}{k}\right)\times opt+1$ approximate solution ok. So, these are long jobs and short jobs are the other jobs. and then we use this fact that if I have a schedule for long jobs which is opt then we run the list scheduling algorithm for short jobs from the optimal schedule of long jobs. So, if we have a schedule for long jobs with make span say M.

then running list scheduling let us call this schedule say S running list scheduling for small jobs. from S gives a schedule with makespan at most. So, in one case we have if we do not need. So, in one case this could be at most m, because if after scheduling all the short jobs if still the long there is a long job which ends last then that makespan will remain m. or if there is a short job which finishes at last, then because short job they are processing time is less than $\frac{opt}{k}$, because opt is greater than equal to k.

average load of the machine. So, because processing time of short jobs is less than $\frac{opt}{k}$

in this case we have a schedule of make span at most $\left(1+\frac{1}{k}\right)\times opt$. and then we computed an optimal schedule for long jobs, hence m becomes opt and then with this we got a $1+\frac{1}{k}$ factor approximation algorithm. And we use the brute force technique which runs in polynomial time assuming k is constant only if the number of machines m is constant. The idea to generalize this algorithm to a compute a schedule which is at most $\left(1+\frac{1}{k}\right)\times opt$ it not be optimal and that will be enough.

So, that we will do now. So, the idea is again divide the jobs into 2 sets long jobs and short jobs and we first observe that the following algorithm is enough is enough. What is this algorithm? It given T a target the algorithm B_k outputs a schedule with make span at most $\left(1+\frac{1}{k}\right)\times T$ if there is a schedule or the algorithm B_k outputs a schedule of make span at most $\left(1+\frac{1}{k}\right)\times T$ or it reports that there is no schedule with make span at most T. So, let us let us first see how if I have an algorithm for B_k I can find a $\left(1+\frac{1}{k}\right)\times opt$ factor approximation algorithm for long jobs. So, we know that $T\neq opt$ long we are focusing only on long jobs because of this result ok.

So, opt_{long} is in between these 2 quantities the first one is it is less than the max of we have 2 lower bounds one is average load $\frac{1}{m}\sum_{i=1}^{n} p_i$ and maximum processing time of any job. So, both are lower bounds. So, this will be greater than equal to. so opt will be greater than equal to this because both are lower bound and we have a because of least scheduling algorithm we have a upper bound on summation average load plus the minimum load of any or the maximum processing time of any job $max_{i=1}^{n} p_i$ ok. So, this one we call it upper bound U and this one we have called it lower bound L.

Now, we start we run B_k with $T=\left\lfloor \frac{L+U}{2} \right\rfloor$. So, our algorithm is like doing binary search every iteration we will ensure that whatever be our u we always have a schedule with makespan at most $(1+\frac{1}{k})\times opt$. So, this is the lower bound and this is the upper bound, this is the upper bound U, this is the lower bound L. Of course, this invariant holds at the beginning of the algorithm because of least scheduling algorithm. Least scheduling algorithm outputs a schedule with make span at most U.

So, if so, we run B_k with this particular T, B_k can output two things. B_k outputs a schedule with make span at most $(1+\frac{1}{k})\times T$ then we update U to T. And again you notice that after this update the invariant remains true that we whatever be the value of U we always have a schedule with make span at most $(1+\frac{1}{k})\times U$. Otherwise if B_k outputs that there is no schedule with makespan at most T, then we update L to $(T+1)$ ok. So, basically we are maintaining two invariants, we maintain the following 2 invariants after each update.

the first invariant is opt is always in between L and U and second is we can compute the schedule with makespan at most $(1+\frac{1}{k})\times U$ in time needed to execute B_k ok. Note that this are this these two invariants hold at the beginning of the iteration and also we observe that in each iteration in each iteration $U-L$ strictly decreases. Moreover, the number of iterations is $O(\log(U-L))$ ok. And we also want to maintain the invariant then U at that U and L are integers for that we write ceiling here, because this ceiling is also a bound ok. So, this shows that having an algorithm B_k which outputs a schedule for long jobs which is which takes T as the target make span as input and either reports that there is no schedule with make span at most T or it outputs a schedule with make span at most $(1+\frac{1}{k})\times T$ that is enough.

to have M to schedule the long jobs with time at most $(1+\frac{1}{k})\times opt$. Because, after this many iterations this after which we have $L=U$ when the algorithm terminates because when$L=U$ that means, it should be equal to opt because opt is sandwiched between L and U and we have an algorithm this B_k outputs a schedule with make span at most $(1+\frac{1}{k})\times U$. So, now, we will describe the algorithm B_k and the idea is again the same that we will partition the jobs into long jobs and short jobs. So, now, we are running we are doing the algorithm B_k. So, here also we define long jobs to be those jobs with processing time p_l is greater than equal to $(1+\frac{1}{k})\times T$, here we are given a target make span T and short jobs $p_l<\frac{1}{k}\times T$.

Again let us show that we can it is enough to schedule only long jobs . So, further what we do we schedule we or let me write enough to schedule long jobs with make span at most $(1+\frac{1}{k})\times T$. Why? Suppose I have a schedule with for long jobs under this definition

of long and short which whose make span is at most this, because if we run again least scheduling algorithm for short jobs from a schedule for long jobs whose makespan is at most $(1+\frac{1}{k})\times T$, then the make span of the schedule obtained for all jobs is at most. again think of two cases we start with a schedule for long jobs and we process the short jobs using the least scheduling algorithm. Now, if there is a long job which finishes last then the make span remains same in this case the make span will remain to be $(1+\frac{1}{k})\times T$.

In the other case if the there is a short job which finishes last then by the analysis of the least scheduling algorithm we know that the makespan when look at the time when the last short job which finished the last started at that time at that time all the machines were busy processing the other jobs and hence the makespan is at most opt plus the processing time of the last job which is at most $\frac{T}{k}$. So, this is the max ok. Now, because T is greater than equal to opt this is $(1+\frac{1}{k})\times T$. So, again our reduced goal is to schedule this long jobs whose processing time is $L \geq \frac{T}{k}$. but may not be optimally it is enough to have schedule them with make span at most $(1+\frac{1}{k})\times T$ and the trick is again rounding down the numbers.

So, we round down the processing time of long jobs to nearest multiple of $\frac{T}{k^2}$. ok. So, we schedule only long jobs we are ignoring short jobs because of this. So, we round down this. Now, here let I' be the resulting instance what we will do? We will compute the optimal solution for I' and we can see that opt_I and $opt_{I'}$ cannot vary much.

For that you see we observe that any machine can process at most k long jobs, if opt_I is less than equal to T ok. If opt_I is less So, if there is a schedule with optimal makespan is at most T because the max the processing time of each job is greater than equal to $\frac{T}{k}$. Now, we can see opt_I of course, it is greater than equal to $opt_{I'}$ because the processing time of the jobs has reduced. when I say I here only long jobs are there, but how much it can. So, if you look at the of I' the same schedule how much more time it can take each rounding may have shorten the processing time of a job by at most k^2 and in each machine there can be at most k jobs.

So, this is $\frac{k \times T}{k^2}$. So, this is $opt_{I'}$ which is also less than equal to T if we have this $(1+\frac{1}{k}) \times T$ ok. So, in the next class we will see a beautiful dynamic programming algorithm for computing the optimal schedule for this long jobs ok. So, let us stop here. Thank you.