Approximation Algorithm

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 12

Lecture 59

Lecture 59 : Inapproximability of Scheduling Jobs on Multiple Non-identical Machines

Welcome. So, in this course till now we have mostly been focusing on various algorithmic aspects for a designing an approximation algorithm for NP complete problems. So, in the remaining 2 lectures we will give a very high level overview of inapproximability, how to show that some problem is an inapproximable within some approximation factor of alpha by any polynomial time algorithm. We have already seen some of the proofs for example, for travelling salesman problem we have seen that there is no row factor approximation algorithm for any polynomial time computable function row. And so, we will see more of this in these two lectures. So, our first problem is scheduling jobs on multiple, but non identical machines.

So, scheduling jobs on multiple parallel non identical. machines . Recall that we have seen the job scheduling problem on multiple parallel identical machines for minimizing the makespan of the schedule and we have seen that that problem admits a polynomial time approximation scheme also known as PTAS. But for this problem we will see that there is no than $\frac{3}{2}$ factor polynomial time approximation algorithm for this problem.

So, before that let us formally define the problem. Input n jobs, m machines for or it takes $p_{i,j}$ amount of time for machine j to process job i, goal, schedule these n jobs. non preemptively preemptively. That means, a job once started cannot be paused in the middle it should be allowed to finish on to m machines to minimize the maximum completion time of any job also known as the makespan.

of the schedule ok. So, we will do a reduction from classical 3 dimensional matching problem to this problem which will give us the required inapproximability bound. So, let us recall 3 dimensional matching. 3 dimensional matching problem input 3 sets X, Y and Z each with each of cardinality n and a collection of triples.

is subset of Cartesian product of X, Y, and Z, cardinality S is m. Computational question does there exist $S^{'} \subseteq S$ of cardinality n such that all x, y, and z appear in $S^{'}$. That means, let us define $S^{'}$ of x to be the first coordinate of $x^{'}$. So, this is all $x \in X$ such that there exist a triple $(x, y, z) \in S^{'}$ that means, $S^{'}$ projected on X that should be equal to X. And, similarly $S^{'}$ projected to Y should be equal to Y and $S^{'}$ projected to Z should be equal to Z.

Because, cardinality of $S^{'}$ is n each element of X, Y, and Z should appear exactly once in $S^{'}$ they appear in exactly one triple in $S^{'}$. So, we reduce three-dimensional matching in polynomial time to this scheduling job. So, reducing means we need to take an arbitrary instance of 3 dimensional matching and convert it to an or construct a scheduling instance which are equivalent in the sense that this 3 dimensional matching instance is a yes instance if and only if the scheduling instance is an yes instance. X Y Z and S be an arbitrary instance of 3 dimensional matching 3 D M for short. If cardinality of X which is n is more than m which is cardinality of S, then clearly the is a no instance, in this case we output a trivial no instance.

So, for applying the framework of NP complete reduction or polynomial time many to one reductions, we need to study decision versions. So, we have formulated the three dimensional matching problem as a decision problem. We need to formulate the scheduling problem as well as a decision problem. The way it is formulated is an optimization problem. So, to formulate it as a decision problem.

we have this inputs and another target makespan t and the question is does there exist with makespan being at most t. So, this is the decision version of this scheduling job and with this decision version we will go ahead with this polynomial time many to one reduction because in the definition of many to one reduction we need to output an equivalent instance. And the equivalent instance means that the three dimensional instance three dimensional matching instance is a yes instance if and only if the scheduling instance is a yes instance. So, if the number if the cardinality of X is greater than the number of triples. then there is no hope of having a 3 dimensional matching.

Hence, we hence clearly the 3 dimensional matching instance is a no instance and we need to output an equivalent instance because the in 3 dimensional matching instance is a no instance, we simply output a trivial no instance for this problem for the scheduling job. So, otherwise let us assume that n is less equal to m. We now construct the scheduling instance. So, we have a job we have n jobs. one job per element one job for every element of X.

ok and we have n machines. We have m machines, one machine for every triple in S. Now, let (x, y, z) be a triple in S, then we define that the jobs corresponding to. So, we have 3 n jobs not n jobs 3 n jobs for every element in x, y and z. So, for every triple we define that the jobs corresponding to these elements x, y and z take 1 unit of time on the machine.

corresponding to the triple (x, y, z). Now, you see that if the 3 dimensional matching instance is a yes instance, then there are n triples which covers all elements. Equivalently in the scheduling term, there are n machines which can execute all these 3 n jobs. and each machine runs for exactly 3 units of time this leaves $m-n$ machines free. So, to occupy those machines we have $m-n$ which is greater than equal to 0 jobs these are dummy jobs.

that need 3 units of processing time on every machine. So, what we have observed what we have just discussed is that if the 3 dimensional matching instance is a yes instance, then there is a schedule with makespan 3. Otherwise the makespan of every schedule is at least 4. So, you see that given a scheduling instance it is NP complete to decide whether the mixpan of the optimal schedule is 3 or 4. This shows that we cannot have a better than $\frac{4}{3}$ factor approximation algorithm.

This implies that for the or there is no α there is no polynomial time alpha approximation algorithm for scheduling for every α less than $\frac{4}{3}$ Why? Because if there is a α approximation algorithm which runs in polynomial time, then if the reduced instance of the scheduling problem has a makespan of 3, it is forced to output that schedule. the makespan should be at most 3 times α or less than 3 times α which is strictly less than 4, but all the processing times are integral here. So, for this instance if whenever the schedule has a makespan 3, the algorithm is forced to output the schedule of makespan 3. On other hand, if the makespan of every schedule is at least 4, then the alpha approximation algorithm will output a schedule of makespan greater than 3. So, using that algorithm we can decide whether a scheduling task has a makespan 3 or more which is enough to have a polynomial time algorithm for 3 dimensional matching.

Next what we do we slightly refine this reduction to improve this bound. What we do is that we have 2 n jobs one each for every element of Y and Z. Now, we have we play with dummy jobs. So, for so, we have n dummy jobs. one each for every element of X and let n X.

for $x \in X$ be the number of number of machine number of tuples which contain x. Actually we have not n dummy jobs, n types of dummy job, one type So, what I want is that among this n X machines all but 1 machine are required for processing the dummy jobs of type X. So, we have $n|X|-1$ dummy jobs of type X ok. So, this now we need to decide the processing times. So, for every (x, y, z) the jobs corresponding to y and z take 1 unit of time.

on the machine corresponding to (x, y, z) ok and for if for every $x \in X$ the dummy job of type X takes 2 units of time. on the on every machine corresponding to a tuple (x, y, z). for any $y \in Y$, $z \in Z$ ok. So, for all the tuples where the first component in x in those machines the dummy job of type x takes only 2 units of time. In other machines the dummy job of type X takes infinite amount of time.

we can replace this infinity with some large positive number which is say in this case 3 is enough or let us keep it 3 that is more concrete. 3 amount of time. So, in the next lecture we will see how this reduction gives a $\frac{3}{2}$ better than $\frac{3}{2}$ factor polynomial time inapproximability for the scheduling task ok. So, let us stop here. Thank you.