18.415/6.854 Advanced Algorithms	December 3, 2001

Lecture 21

Lecturer: Michel X. Goemans *Scribe: Yogishwar Maharaj and Catherine Matlon*

1 Polynomial Approximation Schemes

Definition 1 *Polynomial Approximation Scheme (PAS) is a family of approximation algorithms such that $A_\epsilon \in \{A_\epsilon : \epsilon > 0\}$ runs in polynomial time in the size of the input (assume ϵ fixed) and returns a $1 + \epsilon$ approximate solution.*

Definition 2 *A Fully Polynomial Approximation Algorithm (FPAS) is a family of algorithms such that A_ϵ is a $(1 + \epsilon)$-approximation algorithm with running time polynomial in input size and $1/\epsilon$.*

2 Scheduling Problem: $P\|C_{max}$

Definition 3 ***The Scheduling Problem*** *($P\|C_{max}$): Given n jobs and m machines where each job j takes p_j processing time and completes at time c_j, assign jobs to each machine minimizing the time C_{max} for the last machine to terminate its last job.*

$$C_{max} = T^* = min \max_j c_j$$

2.1 The Approach

Definition 4 *A $(1+\epsilon)$ relaxed decision procedure for $P\|C_{max}$ is an algorithm that, given T, either says that there is no schedule with $C_{max} \leq T$ or gives a schedule with $C_{max} \leq T(1+\epsilon)$*

Initially T^* is between L and $2L$, where $L = max(\Sigma\frac{p_j}{m}, \max p_j)$, so let T_1 and T_2 be L and $2L$ respectively. We're now going to do a logarithmic binary search on the possible values for T^* until we are within ϵ of T^*.

Logarithmic Binary Search: If we know that T^* is between T_1 and T_2, the next value we will check is $\sqrt{T_1T_2}$, which is the midpoint of T_1 and T_2 on the logarithmic scale. If our $(1 + \epsilon)$ relaxed decision procedure returns *NO* on $\sqrt{T_1T_2}$, we replace T_2 with $\sqrt{T_1T_2}$ else we replace T_1 with $\sqrt{T_1T_2}$ and continue until we are within ϵ of T^*.

Initially, $\frac{T_2}{T_1} = 2$. After k iterations, $\log T_2 - \log T_1 = 2^{-k}\log 2$. So if we want $\frac{T_2}{T_1} \leq 1+\epsilon'$, $2^k \sim \log 2/\log(1+\epsilon')$, $k \sim \log(\log 2/\log(1+\epsilon'))$. So, with k iterations, where $k = O(\log\frac{1}{\epsilon'})$, we can get T_1 and T_2 with properties: $T_2/T_1 \leq 1+\epsilon'$, there is no schedule with $C_{max} \leq T_1$, and we have a schedule with $C_{max} \leq T_2(1+\epsilon')$ or $T_2(1+\epsilon/2) \leq T_1(1+\epsilon')(1+\epsilon/2) \leq T_1(1+\epsilon)$.

2.2 A Relaxed Decision

Definition 5 *A $(1+\epsilon)$ relaxed decision procedure for $P\|C_{max}$ is an algorithm that, given T, either says that there is no schedule with $C_{max} \leq T$ or gives a schedule with $C_{max} \leq T(1+\epsilon)$*

Remark 1 *In the preceding definition, it is possible that the procedure returns* NO, *when a schedule does exist for* $C_{max} \leq T(1+\epsilon)$.

We will use a relaxed decision procedure to solve the scheduling problem. Suppose that we have a $(1+\epsilon)$-relaxed decision procedure for jobs with $p_j \geq \epsilon T$. Then we do the following:

1. Remove all jobs with $p_j < \epsilon T$.

2. Apply the $(1+\epsilon)$-relaxed decision procedure for the remaining jobs.

3. If the procedure returns NO, we return NO. If we get a YES, use any method to try to add in all of the small jobs without going beyond $T(1+\epsilon)$. If we can, return that schedule else return NO.

   It is clear that if there is no schedule satisfying $C_{max} \leq T$ on some subset of the jobs, then we cannot hope for one on all of the jobs. Also if we cannot include a job $p_i \leq \epsilon T$ then that implies that each machine is busy at time $T(1+\epsilon) - p_i > T$, so there can obviously be no schedule that finishes in time T.

Consider a $(1+\epsilon)$ relaxed decision procedure for the case where $\forall p_j \geq \epsilon T$. We want to *round* p_j to a q_j that is of the form $\epsilon T + k\epsilon^2 T$ for some integer k, that is

$$q_j = \max_{k \in \mathbb{N}} \{\epsilon T | k\epsilon^2 T \leq p_j\}$$

Then p_j satisfies the following inequality: $0 \leq p_j - q_j \leq \epsilon^2 T$. We output in polynomial time a schedule for $\{q_j\}$ with $C_{max} \leq T$ or else say *NO*.

- *NO*: return *NO*.

- *YES*: return schedule. We can do this because $\epsilon T \leq p_j \Rightarrow q_j \geq \epsilon T \Rightarrow$ There are at most $\frac{1}{\epsilon}$ jobs per machine. Therefore C_{max} increases by at most $\frac{1}{\epsilon}(\epsilon^2 T) = \epsilon T$.

Now consider instances in which there are at most P jobs per machine and at most Q different processing times. In the above case, we take $P = \frac{1}{\epsilon}$ and $Q = \frac{1}{\epsilon^2}$. The problem is to find a schedule with $C_{max} \leq T$ or claim that no such schedule exists, in polynomial time.

Let $(r_1, \ldots, r_Q)$ be an assignment of jobs on a single machine. Each r_i is the number of jobs of value p_i in the assignment. Let the space of all valid assignments be

$$R = \{(r_1, \ldots, r_Q) \in \mathbb{N}^Q : \sum_i r_i p_i \leq T\}$$

We define a function $f : \mathbb{N}^Q \to \mathbb{N}$, such that $f(n_1, \ldots, n_Q)$ is the minimum number of machines needed to process n_i jobs of value p_i, $i \in \{1, \ldots, Q\}$ within time T.

$$f(n_1, \ldots, n_Q) = 1 + \min_{r \in R} f(n_1 - r_1, \ldots, n_Q - r_Q)$$

where $0 \leq n_i \leq k_i =$ number of jobs of processing time p_i.

We know that $|R| \leq P^Q$ and $|\{(n_1, \ldots, n_Q\}| \leq n^Q$. By hypothesis, both of these bounds are constant. Therefore the total running time is $O(n^Q R) = O(n^Q P^Q) = O(n^{\frac{1}{\epsilon^2}} \frac{1}{\epsilon}^{\frac{1}{\epsilon^2}})$. This is polynomial for fixed ϵ.