CSE 632: Analysis of Algorithms II **Fall 2017**

Lecture 6 (09/15/2017): Scheduling on Identical Machines with DP

Lecturer: Shi Li *Scribe: Luting Chen*

6.1 Scheduling on Identical Machines

The job scheduling problem is defined as the following: Given n jobs with processing time P_1, P_2, P_3, $\cdots$, $P_n \geq 0$ and we need to schedule these n jobs on m machines with $m \geq 1$,the goal is to divide n jobs into m sets J_1, J_2, $\cdots$, J_m, so as to minimize $max_{i\in[m]} \sum_{j\in J_i} P_j$.

In Lecture 4, this problem is solved by greedy algorithm and here we present that it can also be solved in PTAS(polynomial time approximation scheme). The main idea is we only focus on a subset of longest jobs and compute the optimal solution for this subset and then we extend the partial scheduling solution by using list scheduling on the other remaining jobs.

Important parameters include the following:

- C^*_{max}: cost of the optimal solusion
- T: estimation of C^*_{max} and it's a upper bound of C^*_{max} where $T \geq \frac{\sum_{j=1}^{n} P_j}{m} = \frac{P([n])}{m}$.
- k: a fixed positive integer.
- S: short jobs if $P_j \leq \frac{T}{k}$.
- L: long jobs if $P_j \geq \frac{T}{k}$.
- $A^{'}$: a black box algorithm to schedule jobs only in L with cost$\leq \alpha T$.

Algorithm 1 A

1: Run $A^{'}$ to obtain a schedule only of L with makespan at most αT.
2: Run greedy algorithm on jobs in S, starting from the above schedule. Each time run one short job on the machine with the least workload.

Lemma 6.1 *$A^{'}$ gives a schedule with makespan max $\{\alpha, 1 + \frac{1}{k}\}$.*

Proof: Let's use i^* to present the machine with the largest job load. There are two cases we need to distinguish:

- Case1: the last job on i^* is a long job, then $C_{max} \leq \alpha T$.
- Case2: the last job on i^* is a short job, then $C_{max} \leq \frac{1}{m}P([n]) + \frac{1}{k}T \leq (1 + \frac{1}{k})T$.

■

Algorithm $A^{'}$ gives a near-optimal schedule of only long jobs by rounding input sizes and dynamic programming. We round each long job $j \in L$ by $P^{'}_j = \left\lfloor \frac{P_j}{\frac{T}{k^2}} \right\rfloor \frac{T}{k^2}$ and $P^{'}_j$ is very close to P_j. And there are only k^2 different values for $P^{'}_j$ if no job is longer than T. We could use a

k^2-dimensional vector to describe the inputs, where the ith component specifies the number of long jobs of rounded size equal to $\frac{iT}{k^2}$, with $i \in \{1, 2, \cdots, k^2\}$.

Algorithm 2 $A^{'}$

1: $P_j^{'} = \left\lfloor \frac{P_j}{\frac{T}{k^2}} \right\rfloor \frac{T}{k^2}$
2: So we have $P_j^{'} = c(\frac{T}{k^2})$ where $c \in \{k, k+1, \cdots, k^2\}$(start from k because we only consider long jobs)
3: Let $J(c_1, c_2, \cdots, c_{k^2})$ be a set of jobs, where job c_1 has size $\frac{T}{k^2}$, job c_2 has size $\frac{2T}{k^2}$,$\cdot$, job c_{k^2} has size T.
4: Let $f(c_1, c_2, \cdots, c_{k^2})$ be the minimum number of machines to schedule jobs $J(c_1, c_2, \cdots, c_{k^2})$, so that the maximum makespan is T.
5: Recursion:$f(c_1, c_2, \cdots, c_{k^2}) = 1 + min_{c_1^{'}, c_2^{'}, \cdots, c_{k^2}^{'}, where \sum_{k^2}^{l=1} c_l^{'} \frac{T}{k^2} l \leq T} f(c_1 - c_1^{'}, c_2 - c_2^{'}, \cdots, c_{k^2} - c_{k^2}^{'})$.

For the recursion part in Algorithm $A^{'}$,

- $f(0, 0, 0, \cdots, 0) = 0$
- For c_1 from 0 to k, c_2 from 0 to k, $\cdots$, c_{k^2} from 0 to k,
  if $(c_1, c_2, \cdots, c_{k^2}) = (0, 0, \cdots, 0)$,then $f(c_1, c_2, \cdots, c_{k^2}) = 0$
- else $f(c_1, c_2, \cdots, c_{k^2}) = \infty$.
  For $c_1^{'}$ from 0 to k, $c_2^{'}$ from 0 to k, $\cdots$, $c_{k^2}^{'}$ from 0 to k:
  if $c_1^{'} \frac{T}{k^2} + c_2^{'} \frac{T}{k^2} \cdot 2 + c_3^{'} \frac{T}{k^2} \cdot 3 + \cdots + c_{k^2}^{'} \frac{T}{k^2} \cdot k^2 \leq T$,
  $temp = 1 + f(max\left\{0, c_1 - c_1^{'}\right\}, max\left\{0, c_2 - c_2^{'}\right\}, \cdots, max\left\{0, c_{k^2} - c_{k^2}^{'}\right\})$,
  if $temp < f(c_1, c_2, \cdots, c_{k^2})$,
  $f(c_1, c_2, \cdots, c_{k^2}) = temp$ and $\pi(c_1, c_2, \cdots, c_{k^2}) = (c_1^{'}, c_2^{'}, \cdots, c_{k^2}^{'})$

6.2 Summary

In this section, we have a brief summary of using rounding and dynamic programming to solve job scheduling on identical machines problem.

- The overall basic idea is that we round the input integers so that the size will be smaller and can be solved in polynomial time. We also need to pay attention to rounding errors, which should be under control.
- The dynamic programming part has time complexity $n^{O(k^2)}$.
- If $T \geq max\left\{\frac{P[n]}{m}, P_{max}\right\}$, anything can happen: the algorithm either fails or return a schedule with makespan $(1 + \frac{1}{k})T$.
- If $T < max\left\{\frac{P[n]}{m}, P_{max}\right\}$ is an upper bound on the optimum, the algorithm will always return a schedule with makespan $(1 + \frac{1}{k})T$.

6.3 3 Partition Problem

The problem is the following: we have m machines and 3m jobs. Can we find a scheduling of jobs where each machine has 3 jobs and the same total processing time?

This problem is a classic NP-hard and it cannot be solved in polynomial time.