Web Algorithms

Eng. Fabio Persia, PhD

Polynomial time approximation schemes (PTAS)

Def: An algorithm A for an optimization problem $\pi \in NPO$ is a polynomial time approximation scheme for π if, given an input instance $x \in I_\pi$ and a rational number $\varepsilon > 0$, it returns a *(1+ε)-approximate* solution (for MIN) or *(1-ε)-approximate solution* (for MAX) in time polynomial with respect to the size of the instance x.

Remark. The time complexity can be exponential in $\frac{1}{\varepsilon}$.

Example:

Complexity $O\left(n^{\frac{1}{\varepsilon}}\right)$

The complexity of a PTAS can grow “drammatically” when ε decreases.

Remark: for every fixed value of ε, a PTAS corresponds to a polynomial time *(1+ε)-approximation* algorithm.

Min Multiprocessor Scheduling

- INPUT: set of n jobs P, number of processors h, running time t_j for each $p_j \in P$
- SOLUTION: A "schedule" for P, that is a function $f: P \rightarrow \{1, \ldots, h\}$
- MEASURE: makespan or completion time of f , that is

$$\max_{i \in [1 \ldots h]} \sum_{p_j \in P | f(p_j) = i} t_j$$

Let's recall Graham's greedy alg.

Greedy choice: at every step assign a job to one of the currently least loaded processors

Let us fastly recall the basic steps of the proof of the Graham's approximation ratio.

Recall: $T_i(j)$ completion time processor i at the end of time j, h number of processors.

We made use of the following lower bounds to m^*:

- $m^* \geq T/h$: in any solution at least one processor must have completion time T/h (recall $T=\Sigma_j t_j$)
- $m^* \geq t_j$ for every job p_j : in any solution one of the processors must run p_j

In order to upper bound the value of the returned solution, if k is one of the most loaded processors and p_l is the last job assigned to k, by the greedy choice

$$T_k(l-1) \leq (\Sigma_{j<l} t_j) / h \leq (T-t_l) / h.$$

Thus we can derive the following inequality:

$$m = T_k(n) = T_k(l-1) + t_l \le \frac{T - t_l}{h} + t_l =$$

$$= \frac{T}{h} + \left(\frac{h-1}{h}\right) \cdot t_l \le m^* + \frac{h-1}{h} \cdot m^* = \left(2 - \frac{1}{h}\right) \cdot m^*$$

$T/h \le m^*$,
$t_l \le m^*$

Idea for improvement: decrease t_l as much as possible and find a better ratio exploiting the inequalities::

$$m \le \frac{T}{h} + \left(\frac{h-1}{h}\right) \cdot t_l \le m^* + \left(\frac{h-1}{h}\right) \cdot t_l$$

Modifying the algorithm and/or improving the analysis we saw how to progressively upper bound t_l with $m^*/2$ (3/2-appr.), $m^*/3$ (4/3-appr.).

Let us now show how to let t_l arbitrarily small, that is $\varepsilon \cdot m^*$, yielding an $(1+\varepsilon)$-appr., i.e. a *PTAS*

In order to obtain a PTAS …

Let's try to get t_l as small as possible. We can exploit the following lemma.

Lemma If $t_1,\ldots,t_n$ are ordered non increasingly, then $\forall i,\ 1\leq i\leq n,$

$$t_i \leq \frac{T}{i}$$

Proof: Assume by contradiction that $t_i > \frac{T}{i}$, then

$$t_1 + t_2 + \ldots + t_i \geq i \cdot t_i > i \cdot \frac{T}{i} = T$$

A contradiction, as $T = \sum_{i=1}^{n} t_i$. □

PTAS: underlying idea

- Compute the optimal solution for the first q jobs
- Complete by assigning in a greedy way the remaining jobs
- If we can obtain $t_l \leq \varepsilon \cdot m^*$, then

$$m \leq \frac{T}{h} + \left(\frac{h-1}{h}\right) \cdot t_l < \frac{T}{h} + t_l \leq m^* + \varepsilon \cdot m^* = (1+\varepsilon)m^*$$

- Starting from the previous lemma, it is sufficient to set q in such a way that, since $l>q$,
- This holds for $q = \left\lceil \frac{h}{\varepsilon} \right\rceil$. Indeed, inequalities $t_l \leq \frac{T}{l} \leq \frac{T}{q+1} \leq \varepsilon \cdot \frac{T}{h} \leq \varepsilon \cdot m^*$

  are true if we consider $q \geq \frac{h}{\varepsilon} - 1$

Algorithm PTAS-Scheduling

Begin

Order the jobs non increasingly with respect to the times t_i and let $p_1,\ldots,p_n$ the resulting ordered sequence ($t_1 \geq t_2 \geq t_3 \geq \ldots \geq t_n$).

Compute an optimal schedule f for the first $q = \left\lceil \frac{h}{\varepsilon} \right\rceil$ jobs

For j=$q+1$ to n

Assign p_j to a processor i with minimum $T_i(j-1)$ // i.e. $f(p_j)=i$

Return schedule f.

End

Theorem PTAS-Scheduling always returns a $(1+\varepsilon)$-approximate solution.

Proof. Let $t \leq m^*$ the completion time of the optimal solution for the first q jobs.

If $m \leq t$, that is the greedy phase has not increased the completion time, then the algorithm returns an optimal solution.

If $m > t$, then again denoting by k the most loaded processor at the end of the algorithm and with p_l the last job assigned to k in the greedy phase

$$m = T_k(n) = T_k(l-1) + t_l \leq \frac{T - t_l}{h} + t_l = \frac{T}{h} + \left(\frac{h-1}{h}\right) \cdot t_l < \frac{T}{h} + t_l \leq m^* + \varepsilon \cdot m^* = (1+\varepsilon) \cdot m^*$$

That is

$$\frac{m}{m^*} \leq 1 + \varepsilon \qquad \square$$

Thus the algorithm satisfies the approximation requirement of PTAS

But is it a PTAS?

Complexity

- The initial ordering requires $O(n \cdot \log n)$ time steps;
- The exhaustive search of an optimal solution for the first q jobs requires at most $O\left(h^{\frac{h}{\varepsilon}}\right)$, since there are at most $h^q \approx h^{h/\varepsilon}$ possible solutions (h possible choices for each of the q jobs)
- The last for executes at most n iterations, each requiring $O(h)$.

Therefore, the overall time complexity is $O\left(n \cdot \log n + h^{\frac{h}{\epsilon}} + n \cdot h\right)$

Such a complexity is exponential in *h,* and thus it can be exponential in the input size (for example if $h=n$)

Thus we don't have a PTAS, unless we don't fix *h* to be given constant value ($h=2$, *or* $h=3$, *or* $h=100$, …)

Fixing *h* constant is equivalent to say that *h* does not depend on the input instance, or analogously that is not part of the input. In other words, it corresponds to consider the following problem:

Min ***h***-Processor Scheduling

- INPUT: Set of *n* jobs *P*, a running time t_j for each $p_j \in P$
- SOLUTION: A schedule $f:P \rightarrow \{1,\ldots,h\}$ for *P*
- MEASURE: completion time of *f*

Theorem: PTAS-Scheduling is a PTAS for Min h-Processor Scheduling

Proof: As already seen

Complexity $$O\left(n \cdot \log n + h^{\frac{h}{\epsilon}} + n \cdot h\right)$$

that is polynomial in the input size (but exponential in *1/ε*)

Approximation 1+ε $\square$

Remark: Min *h*-Processor Scheduling with *h=2* coincides with the famous Min Partition problem, that in turn admits a PTAS:

Min Partition

- INPUT: set of objects X;

  an integer positive weight a_i for every $o_i \in X$;

- SOLUTION: a partition of X in two subsets X_1 and X_2 such that $X_1 \cap X_2 = \emptyset$ and $X_1 \cup X_2 = X$;

- MEASURE: $$MAX\left\{\sum_{o_i \in X_1} a_i, \sum_{o_i \in X_2} a_i\right\}$$

Clearly, the objects correspond to jobs, the weights to the running times and the two subsets to the two processors.

It is possible to extend the previous result to obtain a PTAS for the Min Processor Scheduling problem, i.e. for any (non-constant) number *h* of processors

Recall that in the previous approximation proof of the PTAS, denoting by *t* the completion time of the obtained optimal schedule for the first *q* jobs, we had two cases:

1) $m \le t$, namely the stage greedy did not increase the completion time, then the algorithm returns the optimal solution because $t \le m^*$
2) $m > t$, in which case the usual steps for the part greedy are applied

Idea: The demonstration of approximation continues to be valid if, instead of determining the optimum for the first *q* jobs, we determine for them an approximate solution, i.e. such that $t \le (1+\varepsilon)m^*$

Lemma There is a dynamic programming algorithm that determines in polynomial time a scheduling for the first q jobs having completion time $t \leq (1+\varepsilon)m^*$.

Theorem There exists a PTAS for Min Multiprocessor Scheduling.