Web Algorithms

Eng. Fabio Persia, PhD

Fully polynomial time approximation schemes (FPTAS)

Def: An algorithm *A* for an optimization problem $\pi \in NPO$ is a fully 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 both with respect to the size of the instance *x* and in $1/\varepsilon$

Remark. FPTAS maintain a good time complexity when ε decreases

Examples

$$O\left(n \cdot \log n + 2^{\frac{1}{\varepsilon}}\right)$$ is not a FPTAS!

$$O\left(\frac{n \cdot \log n}{\varepsilon^{2}}\right)$$ is a FPTAS

In practice we must be careful to ensure that 1/ε (or a function more than logarithmic) does not appear in an exponent.

FPTAS-Knapsack

We know that Prog-Dyn-Knapsack-Dual has a pseudo-polynomial time complexity

$$O\left(n^2 \cdot p_{\max}\right)$$

where $p_{max} = max\ p_j$ is the maximum profit of an object

If we can approximate the original profits with smaller profits then the complexity is reduced, while still obtaining a good approximation.

Let's have a closer look to this idea ….

IDEA

1. approximate profits to multiples of *k* for a suitable integer parameter *k>0:*

$$p_i' = \left\lfloor \frac{p_i}{k} \right\rfloor k$$

1. If *k* is sufficiently small, the new profits p'_i approximate enough the original profits p_i, thus an optimal solution for the p'_i well approximates the optimal solution for the original profits p_i

1. Scaling all profits p'_i divididing them for *k,* that is letting $p_i' = \frac{\left\lfloor \frac{p_i}{k} \right\rfloor k}{k} = \left\lfloor \frac{p_i}{k} \right\rfloor$ we obtain an equivalent instance with smaller profits

4. The dual dynamic programming algorithm applied to this instance thus yields a smaller complexity

Complexity:

$$O(n^2 \cdot p'_{\max}) = O\left(n^2 \cdot \frac{p_{\max}}{k}\right)$$

Approximation:

Error at most *k* for every chosen object, i.e. at most *n·k* in total.

Therefore $m \geq m^* - n \cdot k$.

Idea: choose *k* sufficiently large to yield a polynomial complexity and sufficiently small to provide a good approximation:

$$k = \left\lfloor \frac{\varepsilon \cdot p_{\max}}{n} \right\rfloor$$

Algorithm FPTAS-Knapsack

```
Begin
    $k = \left\lfloor \frac{\varepsilon \cdot p_{\max}}{n} \right\rfloor.$
    Find the optimal solution for the instance with scaled profits $p_i' = \left\lfloor \frac{p_i}{k} \right\rfloor$
    using the pseudo-polynomial dynamic programming algorithm
    *Progr-Dyn-Knapsack-Dual* and let *S* be the set of the returned
    objects

    Return *S*.

End
```

Complexity

$$O\left(n^2 \cdot p'_{\max}\right) = O\left(n^2 \cdot \frac{p_{\max}}{k}\right) = O\left(\frac{n^2 \cdot p_{\max}}{\frac{\varepsilon \cdot p_{\max}}{n}}\right) = O\left(\frac{n^3}{\varepsilon}\right).$$

Approximation or error

$$\frac{m}{m^*} \ge \frac{m^* - n\cdot k}{m^*} = 1 - \frac{n\cdot k}{m^*} \overset{m^* \ge p_{max}}{\ge} 1 - \frac{n\cdot k}{p_{\max}} \ge 1 - \frac{\dfrac{n\cdot\varepsilon\cdot p_{\max}}{n}}{p_{\max}} = 1 - \varepsilon$$

Therefore

$$\frac{m}{m^*} \ge 1 - \varepsilon\,.$$

Remark

We did not scale the weights, because it is not guaranteed that switching back to the original weights the solution remains feasible (it might exceed the total weight b).

Lemma: $m \geq m^* - n \cdot k.$

Proof: Let $S=\{ o_{j1},\ldots,o_{jh} \}$ and $S^*=\{ o_{i1},\ldots,o_{il} \}$ be an optimal solution, then:

$$m=p_{j1}+\ldots+p_{jh} \quad and \quad m^*=p_{i1}+\ldots+p_{il}$$

Assume by contradiction that $m < m^* - n{\cdot}k$, then

$$\left\lfloor \frac{p_{i_1}}{k} \right\rfloor + \ldots + \left\lfloor \frac{p_{i_l}}{k} \right\rfloor \geq \left(\frac{p_{i_1}}{k} - 1 \right) + \ldots + \left(\frac{p_{i_l}}{k} - 1 \right) =$$

By hypothesis
$m < m^* - n{\cdot}k$

$$= \frac{p_{i_1} + \ldots + p_{i_l}}{k} - l \geq \frac{p_{i_1} + \ldots + p_{i_l}}{k} - n = \frac{m^*}{k} - n >$$

$$> \frac{m + n \cdot k}{k} - n = \frac{m}{k} = \frac{p_{j_1} + \ldots + p_{j_h}}{k} \geq \left\lfloor \frac{p_{j_1}}{k} \right\rfloor + \ldots + \left\lfloor \frac{p_{j_h}}{k} \right\rfloor .$$

A CONTRADICTION, as S^* would be a solution strictly better than S for the instance with scaled profits, thus contradicting the optimality of *Progr-Dyn-Knapsack-Dual* for the instance with scaled profits.

Theorem: FPTAS-Knapsack is a FPTAS for Max 0-1 Knapsack

Proof:

Complexity:

$$O\left(\frac{n^2 \cdot p_{\max}}{k}\right) = O\left(\frac{n^2 \cdot p_{\max}}{\frac{\varepsilon \cdot p_{\max}}{n}}\right) = O\left(\frac{n^3}{\varepsilon}\right).$$

Approximation:

By the previous lemma

$$\frac{m}{m^*} \geq \frac{m^* - n \cdot k}{m^*} = 1 - \frac{n \cdot k}{m^*} \geq 1 - \frac{n \cdot k}{p_{\max}} \geq 1 - \frac{n \cdot \frac{\varepsilon \cdot p_{\max}}{n}}{p_{\max}} = 1 - \varepsilon .$$

□

Is it possible to reduce the time complexity?

1. Complexity dynamic programming

$$O\left(n \cdot \sum_j p_j\right) = O(n \cdot n \cdot p_{\max})$$

and with scaled profits
$$O\left(n \cdot \sum_j \frac{p_j}{k}\right) = O\left(n \cdot \frac{n \cdot p_{\max}}{k}\right).$$

2. Approximation

$$1 - \frac{n \cdot k}{m^*} \geq 1 - \frac{n \cdot k}{p_{\max}}$$

Remark: p_{max} and $n{\cdot}p_{max}$ respectively are a lower and an upper bound for m^*, that is $p_{max} \leq m^* \leq n{\cdot}p_{max}$

Remark: p_{max} can be much lower than m^* and $n \cdot p_{max}$ can be much bigger!!

Question: is it possible to provide better bounds to m^*?

Observation:

1. Using a better lower bound for m^*, keeping the same k we get a better approximation or analogously increasing k (reducing the complexity) we maintain the same approximation

1. Using a better upper bound for m^*, keeping the same k we get a better complexity or analogously reducing k (improving the approximation) we maintain the same complexity

How to improve the bounds for m^*?

- Recall the ½-approximation Modified-Greedy algorithm and let m_{mg} the measure of its returned solution. By definition of approximation and as $m_{mg} \leq m^*$,

$$m_{mg} \leq m^* \leq 2 \cdot m_{mg}.$$

Thus, by considering $P' = \left\lceil \frac{2m_{mg}}{k} \right\rceil$ as maximum profit in

in *Progr-Din-Knapsack-Dual (i.e. P' is the number of columns in the table of Progr-Din-Knapsack-Dual),* we get a better complexity…

Setting the parameter

$$k = \left\lfloor \frac{\varepsilon \cdot m_{mg}}{n} \right\rfloor$$

and running the modified *Progr-Din-Knapsack-Dual* on the instance with scaled profits

$$p_i' = \left\lfloor \frac{p_i}{k} \right\rfloor$$

we achieve the following time approximation and complexity …

Complexity

$$O\left(\frac{n \cdot m_{mg}}{k}\right) = O\left(\frac{n \cdot m_{mg}}{\frac{\varepsilon \cdot m_{mg}}{n}}\right) = O\left(\frac{n^2}{\varepsilon}\right).$$

Approximation

$$\frac{m}{m^*} \geq 1 - \frac{n \cdot k}{m^*} \geq 1 - \frac{n \cdot k}{m_{mg}} \geq 1 - \frac{n \cdot \frac{\varepsilon \cdot m_{mg}}{n}}{m_{mg}} = 1 - \varepsilon .$$

New FPTAS-Knapsack

```
Begin
  Compute $m_{mg}$ executing algorithm Modified-Greedy
        and let $k = \left\lfloor \frac{\varepsilon \cdot m_{mg}}{n} \right\rfloor$
    Find the optimal solution for the instance with scaled profits $p_i' = \left\lfloor \frac{p_i}{k} \right\rfloor$
    using the modified Progr-Dyn-Knapsack-Dual and let S be
   the set of the returned objects

    Return S.
End
```

Question: can we avoid the computation of *P'* inside the modified *Progr-Dyn-Knapsack-Dual*?