Lecture 4.4

dinsdag 12 september 2023 14:03

In the first step of constructing a PTAS for the knapsack problem, we need to replace the value of each item x_i by a new value, $value^*(x_i)$, which is a "small" integer. A trivial approach to replacing the value of each item by a new "small" integer value would be to simply round all values up. However, this approach does not work, as:

- The values can be too large *(e.g. such that the running time becomes exponential in the number of inputs)*;
- This gives no control over the approximation ratio.

For a better approach, we round each value to the next multiple of Δ, for a suitable Δ.
After that, we divide by Δ to get "small" integers. All in all, we get that $value^*(x_i) = \left\lceil \frac{value(x_i)}{\Delta} \right\rceil$.

To pick a suitable Δ, we consider that we need that $Alg \geq (1 - \epsilon) \cdot OPT = OPT - \epsilon \cdot OPT$. In other words, the total 'error' we are allowed to make in our solution is at most $\epsilon \cdot OPT$. Due to the rounding, the error on each item value is at most Δ. *(This will be proven later.)* Given that there are at most n items in any solution, we have that $total\ error \leq n \cdot \Delta$. This means that Δ should be picked such that $n \cdot \Delta \leq \epsilon \cdot OPT$. One issue with setting this to an equality is that the value of OPT is unknown. Instead, we can use a lower bound on OPT.

Thus, in the algorithm, we do:

1. $LB \leftarrow \max_{x_i \in X} value(x_i)$ assuming $weight(x_i) \leq W$ for all $x_i \in X$; otherwise, remove violating items from the set first, since they are 'useless' anyway.
2. $\Delta \leftarrow \frac{\epsilon}{n} \cdot LB$
3. For each $x_i \in X$, set $value^*(x_i) \leftarrow \left\lceil \frac{value(x_i)}{\Delta} \right\rceil$.
   Let X^* be the set of items with these modified values and with the same weights as before.
4. $S \leftarrow IntValueKnapsack(X^*, W)$
5. Return subset S of items computed in step 3, but with original values.