Chapter 7 Approximation Algorithms Knapsack Approximation

Algorithm Theory WS 2012/13

Fabian Kuhn

Knapsack



- n items 1, ..., n, each item has weight $w_i > 0$ and value $v_i > 0$
- Knapsack (bag) of capacity W
- Goal: pack items into knapsack such that total weight is at most
 W and total value is maximized:

$$\max \sum_{i \in S} v_i$$

s.t. $S \subseteq \{1, ..., n\}$ and
$$\sum_{i \in S} w_i \leq W$$

• E.g.: jobs of length w_i and value v_i , server available for W time units, try to execute a set of jobs that maximizes the total value

Knapsack: Dynamic Programming Alg.



We have shown:

- If all item weights w_i are integers, using dynamic programming, the knapsack problem can be solved in time O(nW)
- If all values v_i are integers, there is another dynamic progr. algorithm that runs in time $O(n^2V)$, where V is the max. value.

Problems:

- If W and V are large, the algorithms are not polynomial in n
- If the values or weights are not integers, things are even worse (and in general, the algorithms cannot even be applied at all)

Idea:

Can we adapt one the algorithms to at least compute an approximate solution?





- The algorithm has a parameter $\varepsilon > 0$
- We assume that each item alone fits into the knapsack
- We solve the problem with values $\widehat{v_i}$ and weights $\underline{w_i}$ using dynamic programming in time $O(n^2(\widehat{V}))$

Theorem: The described algorithm runs in time $O(n^3/\varepsilon)$.

Proof:

$$\widehat{V} = \max_{1 \le i \le n} \widehat{v_i} = \max_{1 \le i \le n} \left[\frac{v_i n}{\varepsilon V} \right] = \left[\frac{v_i}{\varepsilon V} \right] = \left[\frac{n}{\varepsilon} \right]$$



Theorem: The approximation algorithm computes a feasible solution with approximation ratio at most $1 + \varepsilon$.

Proof:

Define the set of all feasible solutions

$$\underbrace{S} := \left\{ \underline{S} \subseteq \{1, \dots, n\} : \sum_{i \in S} w_i \leq W \right\}$$

- Let \widehat{S}^* be an optimal solution and \widehat{S} be the solution computed by the approximation algorithm.
- We have

$$S^* = \max_{\underline{S} \in S} \sum_{i \in S} v_i$$
, $\hat{S} = \max_{\underline{S} \in S} \sum_{S \in S} \widehat{v}_i$

• Hence, \hat{S} is a feasible solution



Theorem: The approximation algorithm computes a feasible solution with approximation ratio at most $1 + \varepsilon$.

Proof:

Because every item fits into the knapsack, we have

$$\forall i \in \{1, \dots, n\}: \underline{v_i} \leq \sum_{j \in S^*} v_j$$

For the solution of the algorithm, we get

$$\widehat{v_i} \triangleq \left\{ \frac{v_i n}{\varepsilon V} \right\} \implies v_i \leq \left(\frac{\varepsilon V}{n} \right) \cdot \widehat{v_i}$$

Therefore

$$\sum_{i \in S^*} v_i \leq \frac{\varepsilon V}{n} \sum_{i \in S^*} \widehat{v_i} \leq \frac{\varepsilon V}{n} \cdot \sum_{i \in \hat{S}} \widehat{v_i} \leq \frac{\varepsilon V}{n} \cdot \sum_{i \in \hat{S}} \left(\frac{v_i n}{\varepsilon V} + 1\right)$$
ratue of opt. solution

(orig. problem)



Theorem: The approximation algorithm computes a feasible solution with approximation ratio at most $1 + \varepsilon$.

Proof:

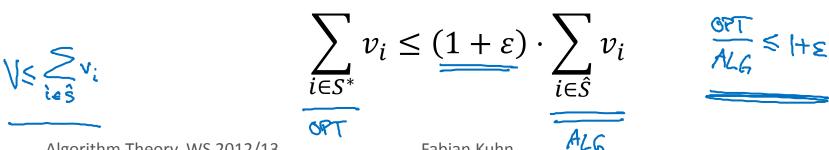
We have

$$\sum_{i \in S^*} v_i \le \frac{\varepsilon V}{n} \cdot \sum_{i \in S^*} \widehat{v_i} \le \frac{\varepsilon V}{n} \cdot \sum_{i \in \hat{S}} \widehat{v_i} \le \frac{\varepsilon V}{n} \cdot \sum_{i \in \hat{S}} \left(\frac{v_i n}{\varepsilon V} + 1\right)$$

Therefore

$$\sum_{i \in S^*} v_i \le \sum_{i \in \hat{S}} v_i + \frac{\varepsilon V}{n} \cdot |\hat{S}| \le \varepsilon V + \sum_{i \in \hat{S}} v_i$$

Because *V* is a lower bound on the optimal solution:



Approximation Schemes $O(ph_0, 2^*)$





- For every parameter $\varepsilon > 0$, the knapsack algorithm computes a $(1+\varepsilon)$ -approximation in time $O(n^3/\varepsilon)$.
- For every fixed ε , we therefore get a polynomial time approximation algorithm
- An algorithm that computes an $(1 + \varepsilon)$ -approximation for every $\varepsilon > 0$ is called an approximation scheme.
- If the running time is polynomial for every fixed ε , we say that the algorithm is a polynomial time approximation scheme (PTAS)
- If the running time is also polynomial in $1/\varepsilon$, the algorithm is a fully polynomial time approximation scheme (FPTAS)
- Thus, the described alg. is an FPTAS for the knapsack problem