Chapter 8
Approximation Algorithms

Algorithm Theory
WS 2019/20

Fabian Kuhn

Knapsack

- n items $1, \ldots, n$, 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:

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

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

- 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 of the algorithms to at least compute an approximate solution?

Approximation Algorithm

- The algorithm has a parameter $\varepsilon > 0$
- We assume that each item alone fits into the knapsack
- We define:

$$V := \max_{1 \le i \le n} v_i, \qquad \forall i: \widehat{v_i} := \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil, \qquad \widehat{V} := \max_{1 \le i \le n} \widehat{v_i}$$

- We solve the problem with integer values $\widehat{v_i}$ and weights w_i using dynamic programming in time $O(n^2 \cdot \widehat{V})$

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

Proof:

$$\widehat{V} = \max_{1 \le i \le n} \widehat{v_i} = \max_{1 \le i \le n} \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil = \left\lceil \frac{Vn}{\varepsilon V} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil$$

Approximation Algorithm

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

Proof:

- Define the set of all feasible solutions (subsets of $[n]$)

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

- $v(S)$: value of solution S w.r.t. values $v_1, v_2, \dots$
  $\hat{v}(S)$: value of solution S w.r.t. values $\hat{v}_1, \hat{v}_2, \dots$
- S^*: an optimal solution w.r.t. values $v_1, v_2, \dots$
  $\hat{S}$: an optimal solution w.r.t. values $\hat{v}_1, \hat{v}_2, \dots$

- Weights are not changed at all, hence, $\hat{S}$ is a feasible solution

Approximation Algorithm

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

Proof:

- We have

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

- Because every item fits into the knapsack, we have

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

- Also: $\widehat{v_i} = \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil \implies v_i \leq \frac{\varepsilon V}{n} \cdot \widehat{v_i},$ and $\widehat{v_i} \leq \frac{v_i n}{\varepsilon V} + 1$

Approximation Algorithm

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

Proof:

- We have

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

- Therefore

$$v(S^*) = \sum_{i \in S^*} v_i \leq \frac{\varepsilon V}{n} \cdot \left|\hat{S}\right| + \sum_{i \in \hat{S}} v_i \leq \varepsilon V + v\left(\hat{S}\right)$$

- We have $v(S^*) \geq V$ and therefore

$$\boldsymbol{(1 - \varepsilon) \cdot v(S^*) \leq v\left(\widehat{S}\right)}$$

Approximation Schemes

- 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