Algorithm Theory

Chapter 8
Approximation Algorithms

Part IV:
Knapsack Approximation Scheme

Fabian Kuhn

Knapsack

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

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

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

- 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 saw two algorithms for the knapsack problem:

- 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 $0 < \varepsilon < 1$
- We assume that each item by itself fits into the knapsack
- We define:

$$V \coloneqq \max_{1 \leq i \leq n} v_i, \qquad \forall i: \hat{v}_i \coloneqq \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil, \qquad \hat{V} \coloneqq \max_{1 \leq i \leq n} \hat{v}_i$$

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

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

Proof:

$$\hat{V} = \max_{1 \leq i \leq n} \hat{v}_i = \max_{1 \leq i \leq n} \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil = \left\lceil \frac{V n}{\varepsilon V} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil$$

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 $[n]$)

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

$$v(S) := \sum_{i \in S} v_i$$

$$\hat{v}(S) := \sum_{i \in S} \hat{v}_i$$

- $v(S)$: 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 $\hat{v}_1, \hat{v}_2, \dots$

$$S^* := \underset{S \in \mathcal{F}}{\operatorname{argmax}}\, v(S)$$

$$\hat{S} := \underset{S \in \mathcal{F}}{\operatorname{argmax}}\, \hat{v}(S)$$

- 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

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

- Because every item fits into the knapsack by itself, we have

$$\forall i \in \{1, \dots, n\}\colon\ v_i \leq V \leq v(S^*)$$

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

$\hat{v}_i \geq \frac{v_i n}{\varepsilon V}$

$\left\lceil \frac{v_i n}{\varepsilon V} \right\rceil \leq \frac{v_i n}{\varepsilon V} + 1$

Approximation Algorithm

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

Proof:

- We have

$v_i \leq \frac{\varepsilon V}{n} \cdot \hat{v}_i$ | $\hat{v}(S^*) \leq \hat{v}(\hat{S})$ | $\hat{v}_i \leq \frac{v_i n}{\varepsilon V} + 1$

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

- Therefore

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

$|\hat{S}| \leq n$ | $= v(\hat{S})$ | $\leq \varepsilon \cdot v(S^*)$

- We have $v(S^*) \geq V$ and therefore $\varepsilon V \leq \varepsilon \cdot v(S^*)$:

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

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 $(1 \pm \varepsilon)$-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