Chapter 8
Approximation Algorithms

Algorithm Theory
WS 2017/18

Fabian Kuhn

Approximation Ratio

An approximation algorithm is an algorithm that computes a solution for an optimization with an objective value that is provably within a bounded factor of the optimal objective value.

Formally:

- $\mathrm{OPT} \geq 0$: optimal objective value
  $\mathrm{ALG} \geq 0$: objective value achieved by the algorithm

- **Approximation Ratio α:**

$$\textbf{Minimization:}\ \alpha := \max_{\text{input instances}} \frac{\mathrm{ALG}}{\mathrm{OPT}} \geq 1$$

$$\textbf{Maximization:}\ \alpha := \min_{\text{input instances}} \frac{\mathrm{ALG}}{\mathrm{OPT}} \leq 1$$

$$\mathbb{E}[\alpha] = \frac{\mathbb{E}[ALG]}{OPT}$$

Knapsack

- n items $1, \ldots, n$, each item has weight $w_i > 0$ and value $v_i > 0$
- Knapsack (bag) of capacity W

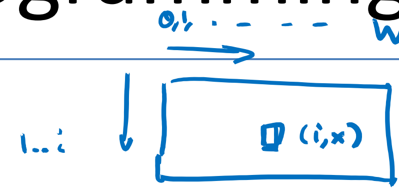
- Goal: pack items into knapsack such that total weight is at most W and total value is maximized:

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

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

- 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.

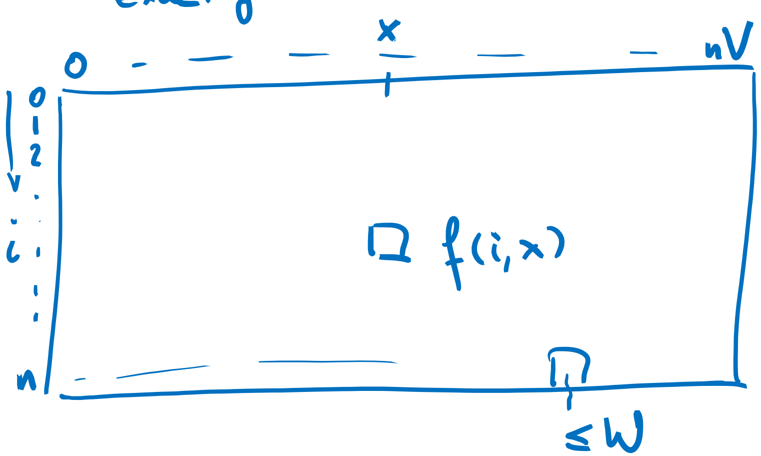


We have seen:

- 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.

$V = \max_i v_i$

$f(i,x)$: min. weight to obtain exactly value x



only items $1,\dots,i$

$f(i,0) = 0$

$f(0,x) = \infty$ (for $x > 0$)

$$f(i,x) = \min \begin{cases} f(i-1,x) \\ f(i-1, x-v_i) + w_i \end{cases}$$

Knapsack: Dynamic Programming Alg.

We have seen:

- 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.

x

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:

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

- We solve the problem with integer values $\hat{v}_i$ and weights w_i using dynamic programming in time $O(n^2 \cdot \hat{V})$
- ~~If solution value $< V$, we take item with value V instead~~

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

$O(n^2 \hat{V})$

Proof:

$$\hat{V} = \max_{1 \le i \le n} \hat{v}_i = \max_{1 \le i \le n} \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil = \left\lceil \frac{Vn}{\varepsilon V} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil \le \left(\frac{n}{\varepsilon} + 1 \right)$$

Approximation Algorithm $\frac{ALG}{OPT} \geq 1-\varepsilon$

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

Proof: $v(\hat{S}) \geq (1-\varepsilon) \cdot v(S^*)$

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

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

computed by dyn. prog.

- $v(S)$: 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

need to show
$v(\hat{S}) \geq (1-\varepsilon) v(S^*)$

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

Proof:

- We have

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

S^*: opt sol. w.r.t. v_i

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

$\hat{S}$: opt sol. w.r.t. $\hat{v}_i$

- Because every item fits into the knapsack, we have

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

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

- 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 $\hat{v}_i \leq \frac{v_i n}{\varepsilon V} + 1$

Approximation Algorithm

$v(\hat{S}) \geq (1-\varepsilon)\, v(S^*)$

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

Proof:

$V \leq v(S^*)$

- We have

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

- Therefore

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

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

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

$\square$

Approximation Schemes

$2^{1/\varepsilon} \cdot n^3$

- For every parameter $\varepsilon > 0$, the knapsack algorithm computes a $(1 \overset{-}{\cancel{+}} \varepsilon)$-approximation in time $O(n^3/\varepsilon)$.

- For every fixed ε, we therefore get a polynomial time approximation algorithm

$(1-\varepsilon)$

- 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