Technische Universität Dresden

International Center for Computational Logic

COMPLEXITY THEORY

Lecture 28: Polynomial-Time Approximation Schemes

Sergei Obiedkov

Knowledge-Based Systems

TU Dresden, 27 Jan 2026

More recent versions of this slide deck might be available.
For the most current version of this course, see
https://iccl.inf.tu-dresden.de/web/Complexity_Theory/en

Approximation Factor

Definition 28.1: Constant-factor approximation $c > 1$

In minimization problems: the solution cost $\leq c \cdot$ the optimal solution cost

In maximization problems: the solution cost $\geq \frac{1}{c} \cdot$ the optimal solution cost

Approximations in polynomial time

Vertex Cover: 2-approximation

Weighted Vertex Cover: 2-approximation

Metric TSP: 3/2-approximation

Low-Diameter Clustering: 2-approximation

General **TSP**: no constant-factor approximation unless P = NP

Clique: no constant-factor approximation unless P = NP

Independent Set: no constant-factor approximation unless P = NP

NP-completeness of KNAPSACK

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items,
each of integral value v_i and weight w_i for $1 \leq i \leq n$;
target value t; and weight limit ℓ

Problem: Is there $T \subseteq I$ such that
$\sum_{i \in T} v_i \geq t$ and $\sum_{i \in T} w_i \leq \ell$?

Theorem 8.8: KNAPSACK is NP-complete.

NP-completeness of KNAPSACK

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items,
each of integral value v_i and weight w_i for $1 \leq i \leq n$;
target value t; and weight limit ℓ

Problem: Is there $T \subseteq I$ such that
$\sum_{i \in T} v_i \geq t$ and $\sum_{i \in T} w_i \leq \ell$?

Theorem 8.8: KNAPSACK is NP-complete.

Proof:

(1) KNAPSACK $\in$ NP: Take T to be the certificate.

(2) KNAPSACK is NP-hard: SUBSET SUM $\leq_p$ KNAPSACK

KNAPSACK: Maximization Version

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items,
each of integral value v_i and weight w_i for $1 \leq i \leq n$;
and weight limit ℓ. Assume $w_i \leq \ell$ for all i.

Problem: Find $T \subseteq I$ such that
$\sum_{i \in T} w_i \leq \ell$ and $\sum_{i \in S} v_i$ is maximal

KNAPSACK: Maximization Version

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items,
each of integral value v_i and weight w_i for $1 \leq i \leq n$;
and weight limit ℓ. Assume $w_i \leq \ell$ for all i.

Problem: Find $T \subseteq I$ such that
$\sum_{i \in T} w_i \leq \ell$ and $\sum_{i \in S} v_i$ is maximal

Example 28.2:

item i	1	2	3	4
value v_i	13	10	6	5
weight w_i	12	3	14	9

- What is an optimal solution for $\ell = 20$?

A Pseudo-Polynomial Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

Initialisation:

- Create an $(\ell + 1) \times (n + 1)$ matrix M
- Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

A Pseudo-Polynomial Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

Initialisation:

- Create an $(\ell + 1) \times (n + 1)$ matrix M
- Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Computation: Assign further $M(w, i)$ to be the largest total value obtainable by selecting from the first i items with weight limit w:

For $i = 0, 1, \ldots, n - 1$, set $M(w, i + 1)$ as

$$M(w, i + 1) := \max \{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\}$$

Here, if $w - w_{i+1} < 0$, we always take $M(w, i)$.

Solution: Take the items contributing to the value in cell $M(\ell, n)$.

Another Pseudo-Polynomial Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n^2 v^*)$, where $v^* = \max_{i \in I} v_i$, using dynamic programming

Initialisation:

- Create an $(V+1) \times (n+1)$ matrix M, where $V = \sum_{i \in I} v_i = O(nv^*)$
- Set $M(v, 0) := 0$ for all $1 \leq v \leq V$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Another Pseudo-Polynomial Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n^2 v^*)$, where $v^* = \max_{i \in I} v_i$, using dynamic programming

Initialisation:

- Create an $(V+1) \times (n+1)$ matrix M, where $V = \sum_{i \in I} v_i = O(nv^*)$
- Set $M(v, 0) := 0$ for all $1 \leq v \leq V$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Computation: Assign further $M(v, i)$ to be the smallest capacity needed to obtain a value $\geq v$ by selecting from the first i items:

For $i = 0, 1, \ldots, n-1$, set $M(v, i+1)$ as $+\infty$ if $\sum_{j=1}^{i+1} v_i < v$ and as

$$M(v, i+1) := \min \{M(v, i),\ M(\max\{0, v - v_{i+1}\}, i) + w_{i+1}\}$$

otherwise.

Solution: Select the maximal v for which $M(v, n) \leq \ell$ and take the items contributing to the value in the cell $M(v, n)$.

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1+\varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1+\varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Idea: Use fewer values for rows.

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1+\varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Idea: Use fewer values for rows.

- For example, choose some b and keep only every bth possible value.

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1 + \varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Idea: Use fewer values for rows.

- For example, choose some b and keep only every bth possible value.
- Alternatively, replace all v_i by $\lceil v_i/b \rceil$ and run the algorithm as is.

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1+\varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Idea: Use fewer values for rows.

- For example, choose some b and keep only every bth possible value.
- Alternatively, replace all v_i by $\lceil v_i/b \rceil$ and run the algorithm as is.
- If $b = \varepsilon v^*/2n$, then the algorithm runs in time

$$O\left(\frac{n^2 v^*}{b}\right) = O\left(\frac{n^2 v^* \cdot 2n}{\varepsilon v^*}\right) = O\left(\frac{1}{\varepsilon} n^3\right).$$

Approximate Algorithm for KNAPSACK

Goal: Transform the pseudo-polynomial exact algorithm into a polynomial approximate algorithm.

- For $0 < \varepsilon \leq 1$, we want to obtain a $(1+\varepsilon)$-approximation with an algorithm whose running time polynomially depends on the input size and on $1/\varepsilon$.

Idea: Use fewer values for rows.

- For example, choose some b and keep only every bth possible value.
- Alternatively, replace all v_i by $\lceil v_i/b \rceil$ and run the algorithm as is.
- If $b = \varepsilon v^*/2n$, then the algorithm runs in time

$$O\left(\frac{n^2 v^*}{b}\right) = O\left(\frac{n^2 v^* \cdot 2n}{\varepsilon v^*}\right) = O\left(\frac{1}{\varepsilon} n^3\right).$$

- The solution returned by the algorithm is feasible: it has weight $\leq \ell$.
- How valuable is it?

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Proof: Let $S \subseteq I$ be our solution and let $S^* \subseteq I$ be an optimal solution.

$$\sum_{i \in S^*} \hat{v}_i \leq \sum_{i \in S} \hat{v}_i \qquad \text{since } S \text{ is optimal for the modified values}$$

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Proof: Let $S \subseteq I$ be our solution and let $S^* \subseteq I$ be an optimal solution.

$$\sum_{i \in S^*} \hat{v}_i \leq \sum_{i \in S} \hat{v}_i \qquad \text{since } S \text{ is optimal for the modified values}$$

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \hat{v}_i b \leq \sum_{i \in S} \hat{v}_i b \leq \sum_{i \in S} (v_i + b) \leq nb + \sum_{i \in S} v_i$$

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Proof: Let $S \subseteq I$ be our solution and let $S^* \subseteq I$ be an optimal solution.

$$\sum_{i \in S^*} \hat{v}_i \leq \sum_{i \in S} \hat{v}_i \qquad \text{since } S \text{ is optimal for the modified values}$$

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \hat{v}_i b \leq \sum_{i \in S} \hat{v}_i b \leq \sum_{i \in S} (v_i + b) \leq nb + \sum_{i \in S} v_i$$

Assume $v_j = v^*$. Then, $v_j = 2nb/\varepsilon = \hat{v}_j b$ and $\sum_{i \in S} \hat{v}_i b \geq \hat{v}_j b = v_j = 2nb/\varepsilon$.

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Proof: Let $S \subseteq I$ be our solution and let $S^* \subseteq I$ be an optimal solution.

$$\sum_{i \in S^*} \hat{v}_i \leq \sum_{i \in S} \hat{v}_i \qquad \text{since } S \text{ is optimal for the modified values}$$

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \hat{v}_i b \leq \sum_{i \in S} \hat{v}_i b \leq \sum_{i \in S} (v_i + b) \leq nb + \sum_{i \in S} v_i$$

Assume $v_j = v^*$. Then, $v_j = 2nb/\varepsilon = \hat{v}_j b$ and $\sum_{i \in S} \hat{v}_i b \geq \hat{v}_j b = v_j = 2nb/\varepsilon$.

$$\sum_{i \in S} v_i \geq \sum_{i \in S} \hat{v}_i b - nb \geq 2nb/\varepsilon - nb = (2/\varepsilon - 1)nb \quad \Rightarrow \quad nb \leq \frac{\sum_{i \in S} v_i}{2/\varepsilon - 1} \leq \frac{\sum_{i \in S} v_i}{1/\varepsilon}$$

Approximate Algorithm for KNAPSACK

Approximate algorithm: Replace every v_i by $\hat{v}_i = \lceil v_i/b \rceil$, where $b = \varepsilon v^*/2n$, and run the second pseudo-polynomial algorithm for KNAPSACK.

Theorem 28.3: The approximate algorithm finds a $(1+\varepsilon)$-approximate solution.

Proof: Let $S \subseteq I$ be our solution and let $S^* \subseteq I$ be an optimal solution.

$$\sum_{i \in S^*} \hat{v}_i \leq \sum_{i \in S} \hat{v}_i \qquad \text{since } S \text{ is optimal for the modified values}$$

$$\sum_{i \in S^*} v_i \leq \sum_{i \in S^*} \hat{v}_i b \leq \sum_{i \in S} \hat{v}_i b \leq \sum_{i \in S} (v_i + b) \leq nb + \sum_{i \in S} v_i$$

Assume $v_j = v^*$. Then, $v_j = 2nb/\varepsilon = \hat{v}_j b$ and $\sum_{i \in S} \hat{v}_i b \geq \hat{v}_j b = v_j = 2nb/\varepsilon$.

$$\sum_{i \in S} v_i \geq \sum_{i \in S} \hat{v}_i b - nb \geq 2nb/\varepsilon - nb = (2/\varepsilon - 1)nb \quad \Rightarrow \quad nb \leq \frac{\sum_{i \in S} v_i}{2/\varepsilon - 1} \leq \frac{\sum_{i \in S} v_i}{1/\varepsilon}$$

$$\sum_{i \in S^*} v_i \leq nb + \sum_{i \in S} v_i \leq \varepsilon \sum_{i \in S} v_i + \sum_{i \in S} v_i = (1+\varepsilon) \sum_{i \in S} v_i$$

□

Polynomial-Time Approximation Scheme

Definition 28.4:

- An algorithm A is an approximation scheme for an optimization problem Π if, on input (I, ε), where I is an instance of Π and $\varepsilon > 0$, it outputs a solution that is a $(1+\varepsilon)$-approximation of an optimal solution for I.
- If, for every fixed ε, the running time of A is bounded by a polynomial in the size of I, then A is a polynomial-time approximation scheme (PTAS) for Π.
- If the running time of A is bounded by a polynomial in the size of I and the value of $1/\varepsilon$, then A is a fully polynomial-time approximation scheme (FPTAS) for Π.

- We have shown that **KNAPSACK** has an FPTAS.

Summary and Outlook

Some NP-hard problems admit polynomial-time constant-factor approximation algorithms.

Some others admit a PTAS, making it possible to get as close to an optimal solution as you want in time that depends polynomially on the size of the input but arbitrarily on the desired approximation factor.

For some problems, it is possible to obtain an FPTAS, whose running time depends polynomially on both the input size and the approximation factor.

What's next?

- Parameterized complexity
- Examinations