Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Approximationsalgorithmen

Vorlesung 9: Approximationsschemata und das Knapsack-Problem

Folien von Joachim Spoerhase und Steven Chaplick

[Vazirani: §8, Williamson & Shmoys: §3.1]

Alexander Wolff Lehrstuhl für Informatik I WS 2018/19

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

A PTAS is called **fully polynomial-time approximation scheme** (FPTAS) if its runtime is polynomial in $|I|$ and $1/\epsilon$.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

A PTAS is called **fully polynomial-time approximation scheme** (FPTAS) if its runtime is polynomial in $|I|$ and $1/\epsilon$.

Example running times:

- $O(n^{1/\epsilon^2}) \rightsquigarrow$
- $O(2^{1/\epsilon} n^4) \rightsquigarrow$
- $O((\frac{1}{\epsilon})^2 n^3) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

A PTAS is called **fully polynomial-time approximation scheme** (FPTAS) if its runtime is polynomial in $|I|$ and $1/\epsilon$.

Example running times:

- $O(n^{1/\epsilon^2}) \rightsquigarrow$ PTAS
- $O(2^{1/\epsilon} n^4) \rightsquigarrow$
- $O((\frac{1}{\epsilon})^2 n^3) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

A PTAS is called **fully polynomial-time approximation scheme** (FPTAS) if its runtime is polynomial in $|I|$ and $1/\epsilon$.

Example running times:

- $O(n^{1/\epsilon^2}) \rightsquigarrow$ PTAS
- $O(2^{1/\epsilon} n^4) \rightsquigarrow$ PTAS
- $O((\frac{1}{\epsilon})^2 n^3) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called **polynomial-time approximation scheme** (PTAS) if it outputs for every input (I, ϵ) with $I \in D_\Pi$ and $\epsilon > 0$ a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \geq (1 - \epsilon) \cdot \text{OPT}$ if Π max problem.
- $\text{obj}_\Pi(I, s) \leq (1 + \epsilon) \cdot \text{OPT}$ if Π min problem,

The runtime of $\mathcal{A}$ is polynomial in $|I|$ for every fixed $\epsilon > 0$.

A PTAS is called **fully polynomial-time approximation scheme** (FPTAS) if its runtime is polynomial in $|I|$ and $1/\epsilon$.

Example running times:

- $O(n^{1/\epsilon^2}) \rightsquigarrow$ PTAS
- $O(2^{1/\epsilon} n^4) \rightsquigarrow$ PTAS
- $O((\frac{1}{\epsilon})^2 n^3) \rightsquigarrow$ FPTAS (and PTAS ;-)

Knapsack Problem

We are given a set $S = \{a_1, \ldots, a_n\}$ of **objects**.
For $i = 1, \ldots, n$, object a_i has **size** $\text{size}(a_i) \in \mathbb{N}^+$ and **profit** $\text{profit}(a_i) \in \mathbb{N}^+$. We are also given a (knapsack) **capacity** $B \in \mathbb{N}^+$. We are looking for a subset of the objects whose **total size** is at most B and whose **total profit** is maximum.

Knapsack Problem

We are given a set $S = \{a_1, \ldots, a_n\}$ of **objects**.
For $i = 1, \ldots, n$, object a_i has **size** size$(a_i) \in \mathbb{N}^+$ and **profit** profit$(a_i) \in \mathbb{N}^+$. We are also given a (knapsack) **capacity** $B \in \mathbb{N}^+$. We are looking for a subset of the objects whose **total size** is at most B and whose **total profit** is maximum.

NP-hard

Pseudopolynomial-Time Algorithm

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits). As usual, $|I|$ denotes the size of the instance $I \in D_\Pi$, where all numbers in I are encoded in **binary**. We use $|I|_{\mathsf{u}}$ to denote the size of I when all numbers in I are encoded in **unary**.

Pseudopolynomial-Time Algorithm

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits). As usual, $|I|$ denotes the size of the instance $I \in D_\Pi$, where all numbers in I are encoded in **binary**. We use $|I|_{\mathsf{u}}$ to denote the size of I when all numbers in I are encoded in **unary**.

- The running time of a polynomial-time algorithm for Π is polynomial in $|I|$.

Pseudopolynomial-Time Algorithm

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits). As usual, $|I|$ denotes the size of the instance $I \in D_\Pi$, where all numbers in I are encoded in **binary**. We use $|I|_{\mathsf{u}}$ to denote the size of I when all numbers in I are encoded in **unary**.

- The running time of a polynomial-time algorithm for Π is polynomial in $|I|$.
- The running time of a **pseudo-polynomial-time algorithm** is polynomial in $|I|_{\mathsf{u}}$.

Pseudopolynomial-Time Algorithm

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits). As usual, $|I|$ denotes the size of the instance $I \in D_\Pi$, where all numbers in I are encoded in **binary**. We use $|I|_{\mathsf{u}}$ to denote the size of I when all numbers in I are encoded in **unary**.

- The running time of a polynomial-time algorithm for Π is polynomial in $|I|$.
- The running time of a **pseudo-polynomial-time algorithm** is polynomial in $|I|_{\mathsf{u}}$.
- The running time of a pseudo-polynomial algorithm may not be polynomial in $|I|$.

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \qquad \leq \text{OPT} \leq$

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$
- For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

- For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

- Let $A[i, p]$ denote the total size of $S_{i,p}$ (let $A[i, p] = \infty$ if no such set exists).

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

- For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

- Let $A[i, p]$ denote the total size of $S_{i,p}$ (let $A[i, p] = \infty$ if no such set exists).

- If all entries of the table $A[\cdot, \cdot]$ are known, we can compute OPT $=$

Pseudo-Polynomial Alg. for KNAPSACK

- $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

- For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

- Let $A[i, p]$ denote the total size of $S_{i,p}$ (let $A[i, p] = \infty$ if no such set exists).

- If all entries of the table $A[\cdot, \cdot]$ are known, we can compute $\text{OPT} = \max\{\, p \mid A(n, p) \leq B \,\}$.

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{$ $,$ $\}$

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{\, A[i, p],$ $\}$

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{\, A[i, p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})] \,\}$

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{\, A[i, p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})] \,\}$
- Running time and space consumption:

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{\, A[i, p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})] \,\}$
- Running time and space consumption:
  $A[\cdot\,, \cdot]$ and OPT can be computed in $O(n^2 P)$ time & space.

Pseudo-Polynomial Alg. for KNAPSACK

- $A[1, p]$ can easily be determined for $p \in \{0, \ldots, nP\}$.
- For $i = 1, \ldots, n$, set $A[i, p] := \infty$ for $p < 0$ and $A[i, 0] = 0$.
- $A[i+1, p] = \min\{\, A[i, p], \text{size}(a_{i+1}) + A[i, p - \text{profit}(a_{i+1})] \,\}$
- Running time and space consumption:
  $A[\cdot\,, \cdot]$ and OPT can be computed in $O(n^2 P)$ time & space.

KNAPSACK can be solved to optimality in $O(n^2 P)$ time.

FPTAS for KNAPSACK by Scaling

Note: $O(n^2 P)$ is polynomial in n – if P is polynomial in n :-)

FPTAS for KNAPSACK by Scaling

Note: $O(n^2 P)$ is polynomial in n – if P is polynomial in n :-)

- FPTAS-Idea: **Scale** profits to polynomial size (as required by the error parameter ϵ).

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

FPTAS for Knapsack by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, $\leq K \cdot$ profit$'(o_i) \leq$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, $\quad \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ $\leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ $= \text{OPT} - nK \leq \text{OPT} - kK \leq K \sum_i$ profit$'(o_i)$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ OPT$- \epsilon P =$ OPT$- nK \leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ OPT$- \epsilon P =$ OPT$- nK \leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

Obs. 2. profit$(S') \geq K \cdot$ profit$'(S') \geq K \cdot \sum_i$profit$'(o_i)$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ OPT$- \epsilon P =$ OPT$- nK \leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

Obs. 2. profit$(S') \geq K \cdot$ profit$'(S') \geq K \cdot \sum_i$profit$'(o_i)$

$\Rightarrow$ profit$(S') \geq$ OPT $- \epsilon P \geq$ OPT $- \epsilon$OPT $= (1-\epsilon) \cdot$ OPT $\square$

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ OPT$- \epsilon P =$ OPT$- nK \leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

Obs. 2. profit$(S') \geq K \cdot$ profit$'(S') \geq K \cdot \sum_i$profit$'(o_i)$

$\Rightarrow$ profit$(S') \geq$ OPT $- \epsilon P \geq$ OPT $- \epsilon$OPT $= (1-\epsilon) \cdot$ OPT $\square$

Thm. 1. KnapsackScaling is an FPTAS for KNAPSACK.
The algorithm runs in $O($ $)$ time. [Ibarra & Kim, '75]

FPTAS for KNAPSACK by Scaling

KnapsackScaling(I,ϵ)
  $K \leftarrow \epsilon P/n$
  profit$'(a_i) := \lfloor$profit$(a_i)/K \rfloor$
  compute optimum solution S' for I w.r.t. profit$'(\cdot)$
  return S'

Lemma. profit$(S') \geq (1-\epsilon) \cdot$ OPT.

Proof. Let $o_1, \ldots, o_k$ be an optimal solution.

Obs. 1. For $i = 1, \ldots, k$, profit$(o_i) - K \leq K \cdot$ profit$'(o_i) \leq$ profit(o_i)

$\Rightarrow$ OPT$- \epsilon P =$ OPT$- nK \leq$ OPT$- kK \leq K \sum_i$ profit$'(o_i)$

Obs. 2. profit$(S') \geq K \cdot$ profit$'(S') \geq K \cdot \sum_i$profit$'(o_i)$

$\Rightarrow$ profit$(S') \geq$ OPT $- \epsilon P \geq$ OPT $- \epsilon$OPT $= (1-\epsilon) \cdot$ OPT $\square$

Thm. 1. KnapsackScaling is an FPTAS for KNAPSACK.
The algorithm runs in $O(n^3/\epsilon)$ time. [Ibarra & Kim, '75]

Strong NP-Hardness

An optimization problem is called **strongly NP-hard**
if it remains NP-hard when numbers are encoded in unary.

Strong NP-Hardness

An optimization problem is called **strongly NP-hard**
if it remains NP-hard when numbers are encoded in unary.

Can the KNAPSACK problem be strongly NP-hard?

Strong NP-Hardness

An optimization problem is called **strongly NP-hard**
if it remains NP-hard when numbers are encoded in unary.

Can the KNAPSACK problem be strongly NP-hard?

Thm. 2. A strongly NP-hard problem has no pseudo-polynomial algorithm unless P = NP.

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon =$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow$ ALG $\leq (1+\epsilon)\mathrm{OPT} <$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ ALG $\leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) =$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ $\mathrm{ALG} \leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) = \mathrm{OPT} + 1.$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ $\mathrm{ALG} \leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow$ $\mathrm{ALG} = \mathrm{OPT}.$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ $\mathrm{ALG} \leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow$ $\mathrm{ALG} = \mathrm{OPT}.$

Running time:

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ ALG $\leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) =$ OPT $+ 1$.

$\Rightarrow$ ALG $=$ OPT.

Running time: $q(|I|, p(|I|_{\mathsf{u}}))$

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ $\mathrm{ALG} \leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow$ $\mathrm{ALG} = \mathrm{OPT}$.

Running time: $q(|I|, p(|I|_{\mathsf{u}}))$, that is,

FPTAS and Pseudo-Polynomial Algorithms

Thm. 3. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and with $\mathrm{OPT}(I) < p(|I|_{\mathsf{u}})$ for all instances I of Π. If Π has an FPTAS then there is a pseudo-polynomial algorithm for Π.

Proof.

Suppose there is an FPTAS for Π (running in $q(|I|, 1/\epsilon)$ time).

Set $\epsilon = 1/p(|I|_{\mathsf{u}})$.

$\Rightarrow$ ALG $\leq (1+\epsilon)\mathrm{OPT} < \mathrm{OPT} + \epsilon p(|I|_{\mathsf{u}}) =$ OPT $+ 1$.

$\Rightarrow$ ALG $=$ OPT.

Running time: $q(|I|, p(|I|_{\mathsf{u}}))$, that is, polynomial in $|I|_{\mathsf{u}}$.

FPTAS and Strong NP-Hardness

Corollary. Let Π be an NP-hard optimization problem that fulfils the requirements of Theorem 3.
If Π is strongly NP-hard then there is no FPTAS for Π unless $\mathrm{P} = \mathrm{NP}$.