Widerlegung der Existens eines Approximationsshemas

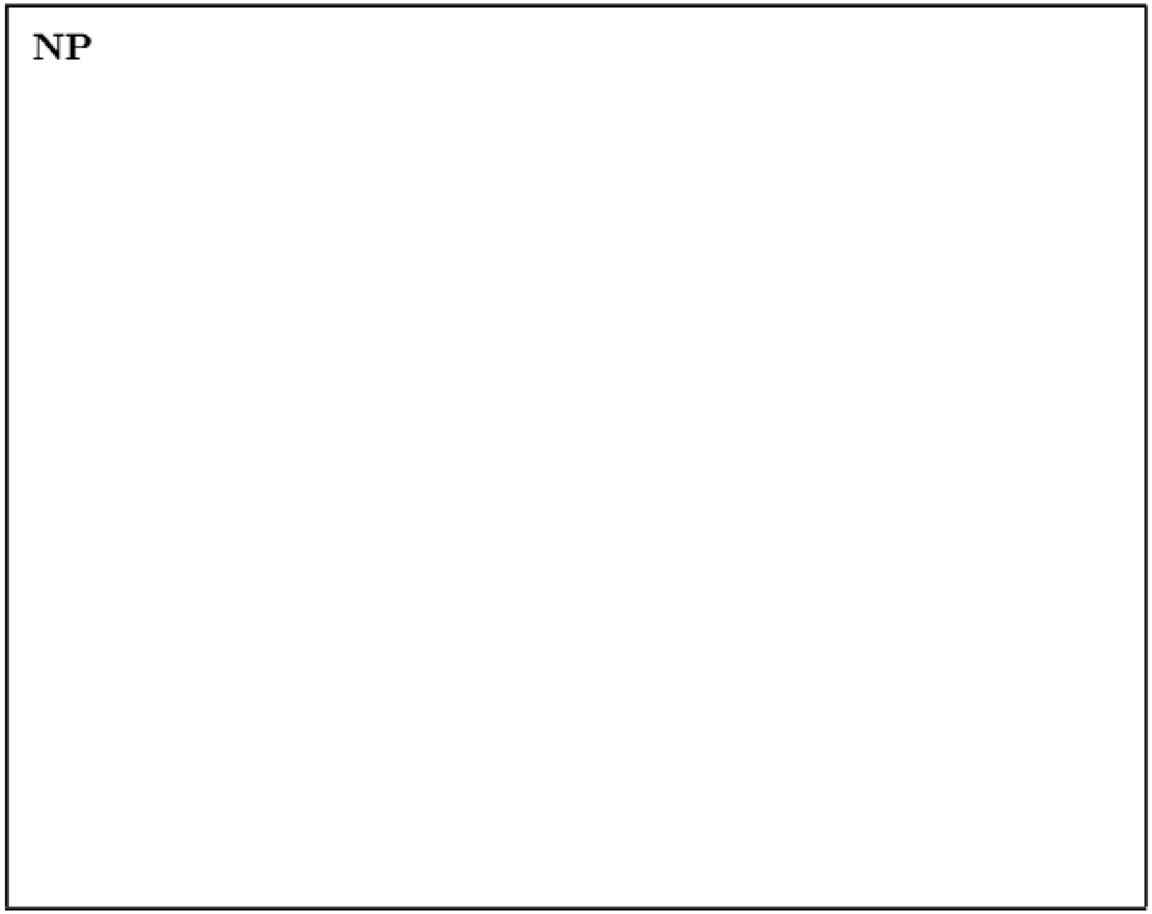
Nikolaus Mutsanas

Universität Karlsruhe

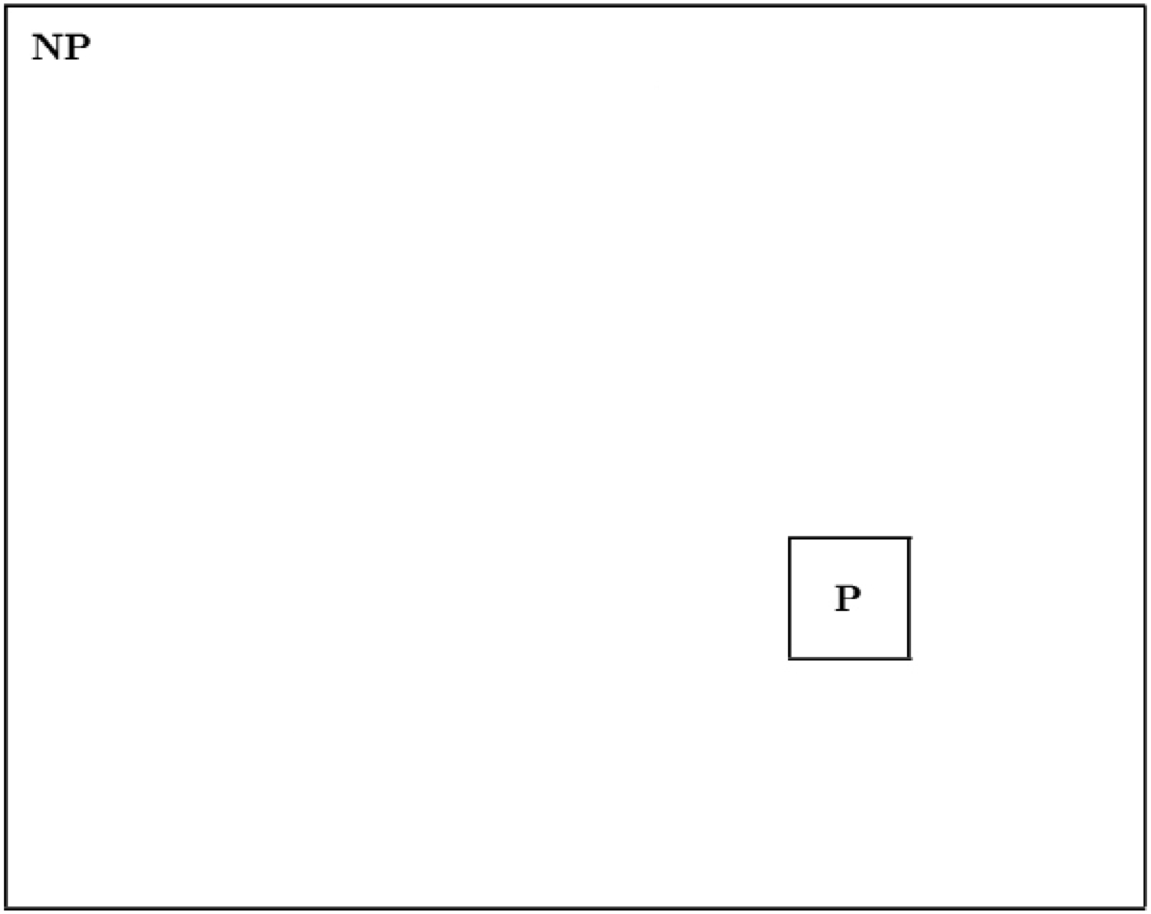
Gliederung

- Komplexitätsklassen
- Widerlegung der Existenz eines FPTAS
  - Beispiel
  - Generalisierung
- Widerlegung der Existenz eines PTAS
  - Die „Lückentechnik“, Anwendung
  - APX-Vollständigkeit
    - L-Reduktion
    - Anwendung

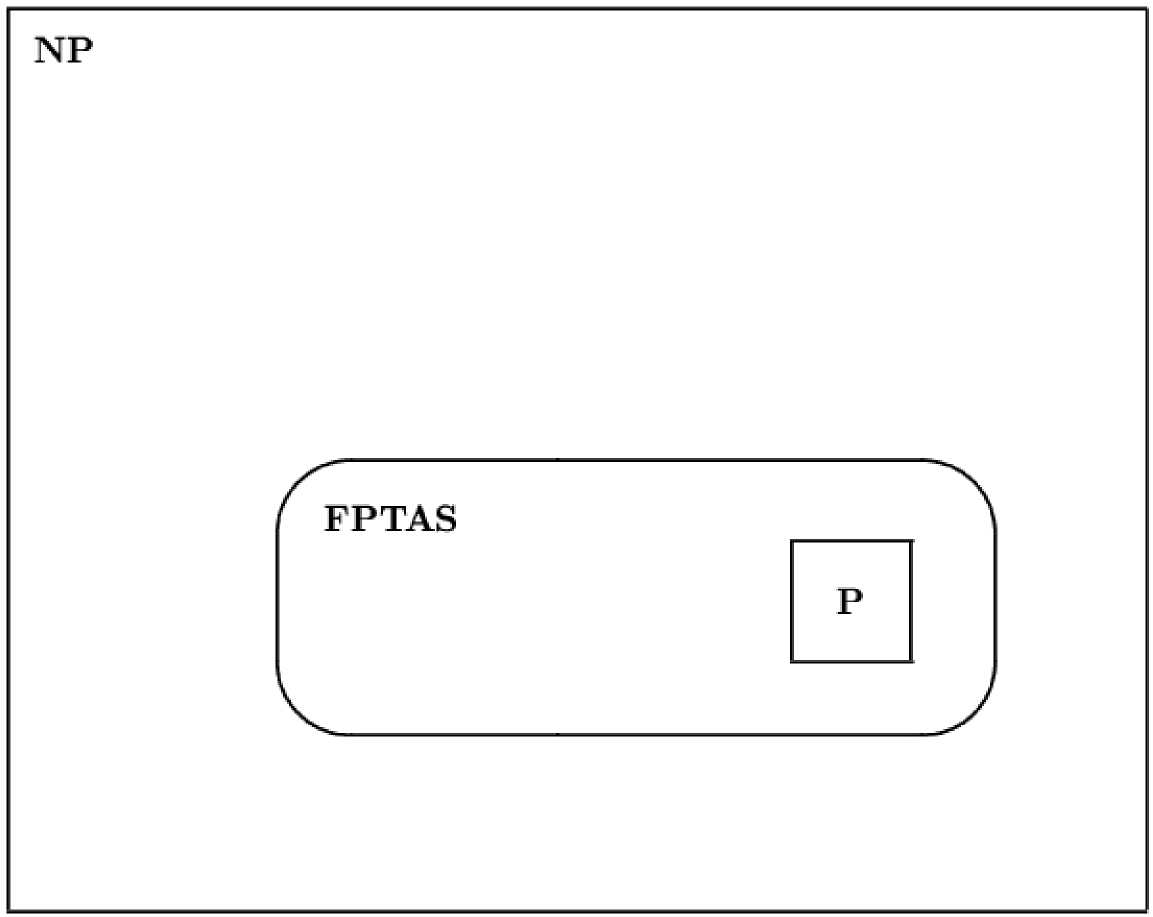
Komplexitätsklassen



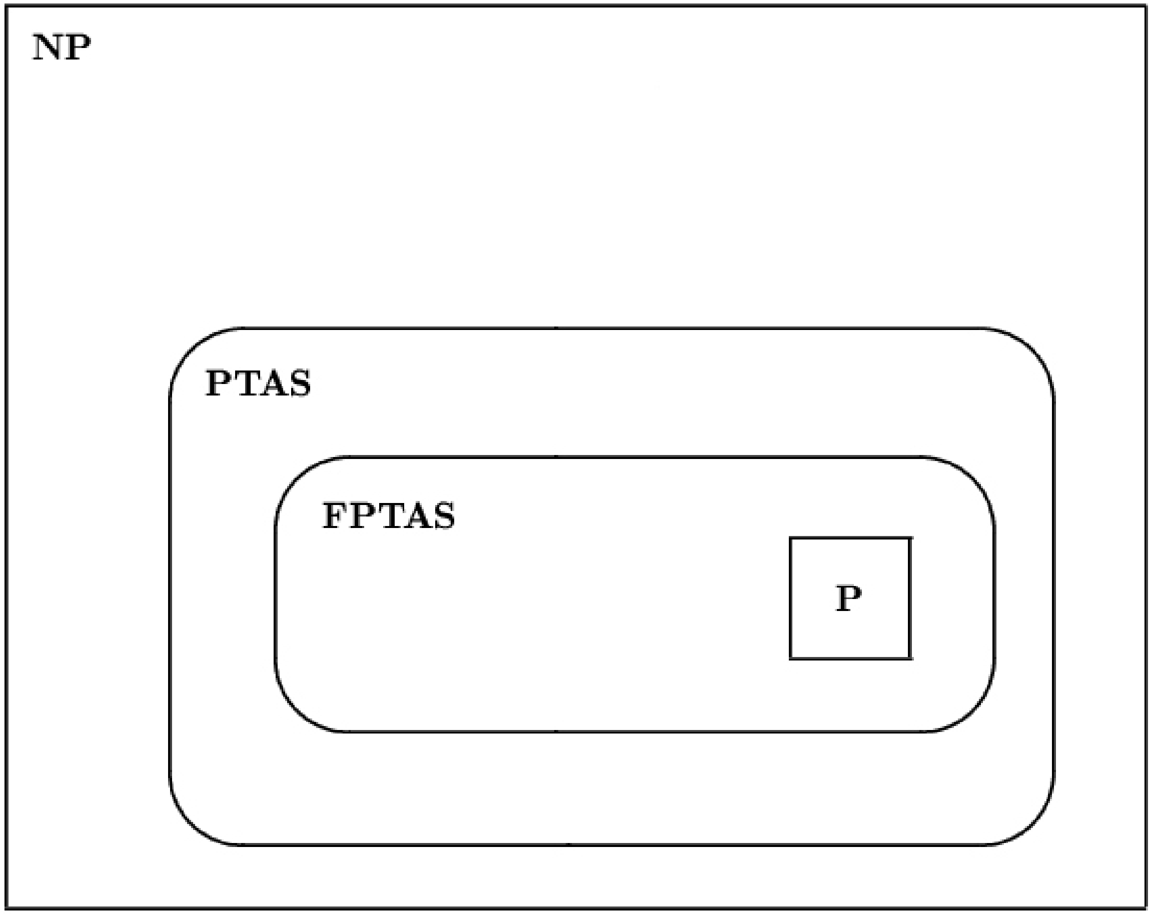
Komplexitätsklassen



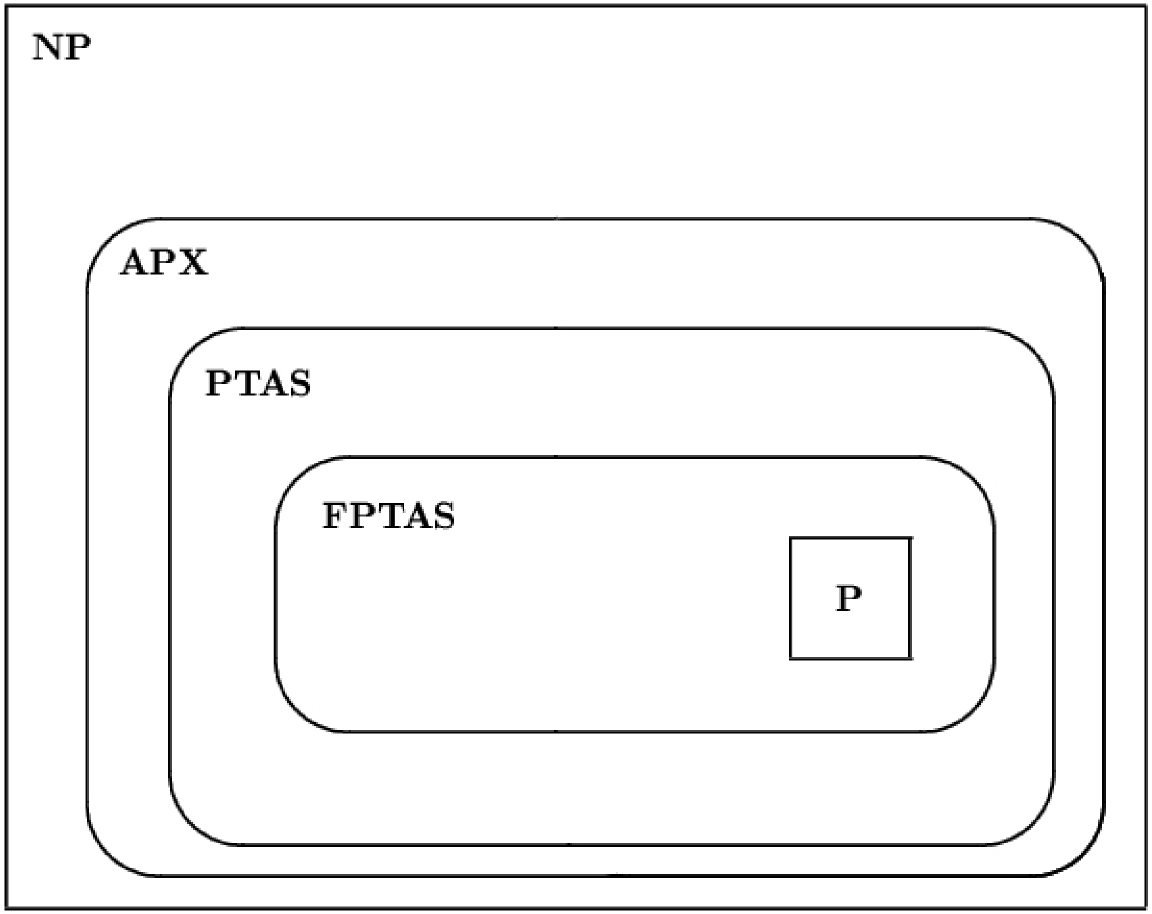
Komplexitätsklassen



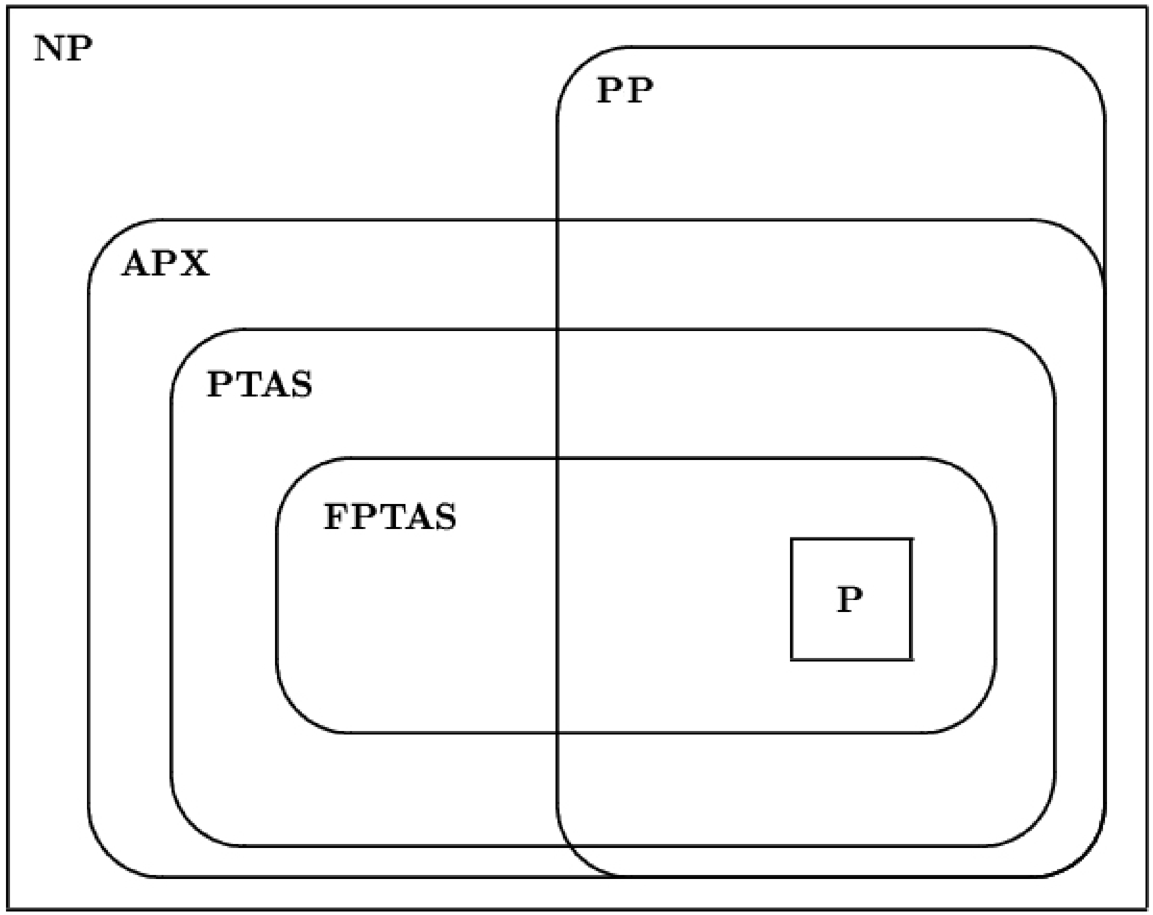
Komplexitätsklassen



Komplexitätsklassen



Komplexitätsklassen



Kodierung

- binär ($n \in \mathbb{N}$ durch $\lfloor \log_2(n) \rfloor + 1$ bits kodiert)

Kodierung

- binär ($n \in \mathbb{N}$ durch $\lfloor \log_2(n) \rfloor + 1$ bits kodiert)
- unär ($n \in \mathbb{N}$ durch n bits kodiert)

Kodierung

- binär ($n \in \mathbb{N}$ durch $\lfloor \log_2(n) \rfloor + 1$ bits kodiert)
- unär ($n \in \mathbb{N}$ durch n bits kodiert)
- $13 = 1101_{\text{binär}} = 1111111111111_{\text{unär}}$

Kodierung

- binär ($n \in \mathbb{N}$ durch $\lfloor \log_2(n) \rfloor + 1$ bits kodiert)
- unär ($n \in \mathbb{N}$ durch n bits kodiert)
- $13 = 1101_{\text{binär}} = 1111111111111_{\text{unär}}$

Für jede Instanz I eines Problems gilt : $|I|_{\text{unär}} > |I|_{\text{binär}}$

Kodierung

- binär ($n \in \mathbb{N}$ durch $\lfloor \log_2(n) \rfloor + 1$ bits kodiert)
- unär ($n \in \mathbb{N}$ durch n bits kodiert)
- $13 = 1101_{\text{binär}} = 1111111111111_{\text{unär}}$

Für jede Instanz I eines Problems gilt : $|I|_{\text{unär}} > |I|_{\text{binär}}$

Komplexität hängt von der Kodierung ab!

- ▷ $\text{Problem}_{\text{binär}}$ NP-schwer, $\text{Problem}_{\text{unär}}$ polynomiell lösbar : *gewöhnlich NP-schwer*
- ▷ $\text{Problem}_{\text{binär}}$ NP-schwer, $\text{Problem}_{\text{unär}}$ NP-schwer : *streng NP-schwer*

Allgemeine Vorgehensweise

Angenommen es gibt einen passenden Approximationsalgorithmus für das Problem X.
$\Rightarrow Z_1$ besitzt einen polynomiellen Algorithmus
$\Rightarrow Z_2$ besitzt einen polynomiellen Algorithmus
$\Rightarrow \ldots$
$\Rightarrow Z_k$ besitzt einen polynomiellen Algorithmus, und es ist nachgewiesen dass Z_k NP-schwer

Widerspruch, falls $\mathrm{P} \neq \mathrm{NP}$

Notation

- ▷ Problem X durch die Menge seiner Instanzen I_X identifiziert.
- ▷ S_{I_X} : Menge der Lösungen der Instanz I_X.
- ▷ $s_{I_X} \in S_{I_X}$ eine Lösung der Instanz I_X
- ▷ $c : S_{I_X} \to \mathbb{R},\ \mathbb{R}_+,\ \mathbb{Z},\ \mathbb{N}$ Kostenfunktion (bzw. Gütefunktion).
- ▷ $\textsc{Opt}(I_X)$: $Kosten$ der Optimalen Lösung von I_X.

$\nexists$ FPTAS

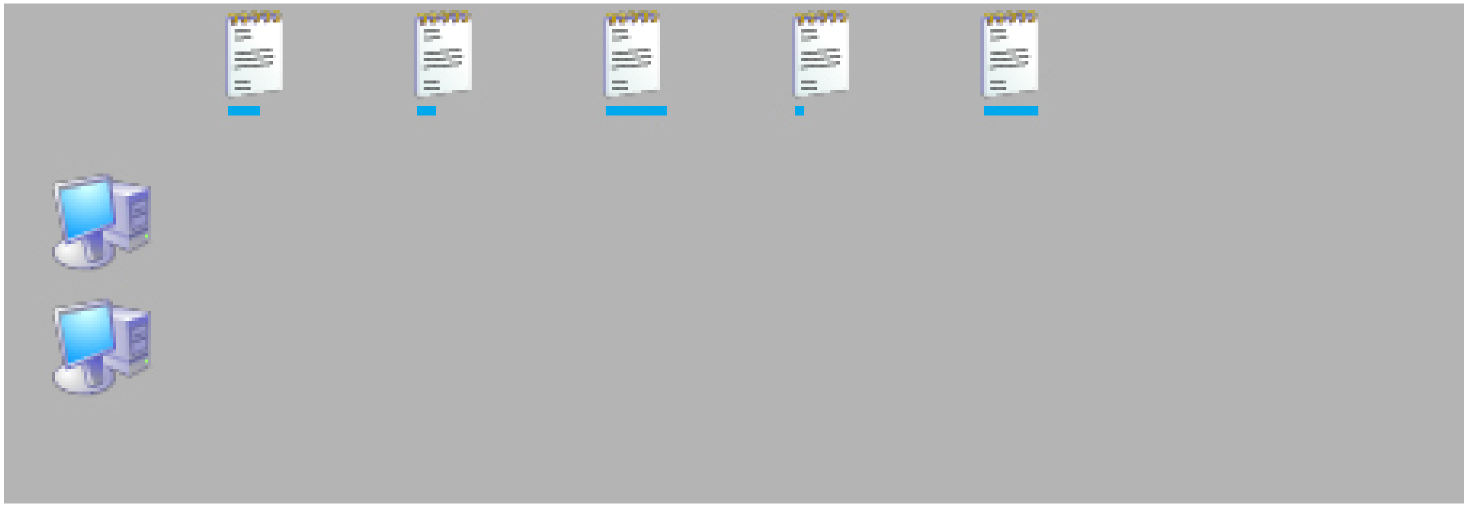
Beispiel:

$P||C_{\max}$: Minimierung der Abarbeitungszeit auf identischen Maschinen:

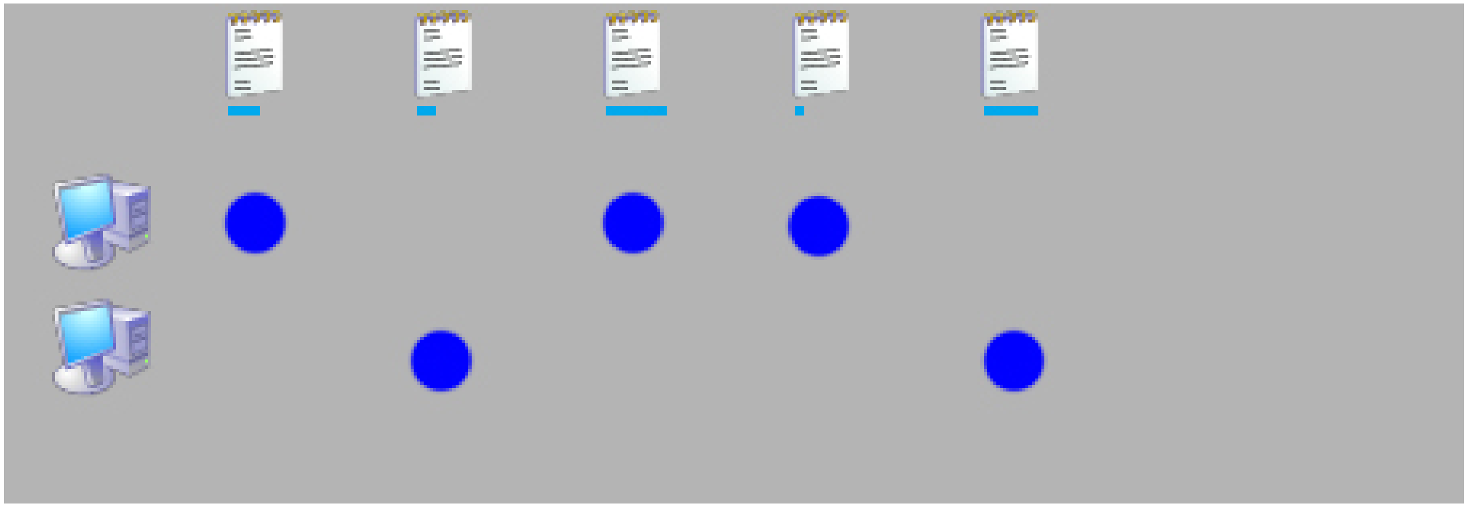
- $m \in \mathbb{N}$ identische Maschinen
- $n \in \mathbb{N}$ Aufträge
- $p_j \in \mathbb{N}$, die Laufzeit des $j-$ten Auftrags

streng NP-schwer (d.h. Kodierung egal).

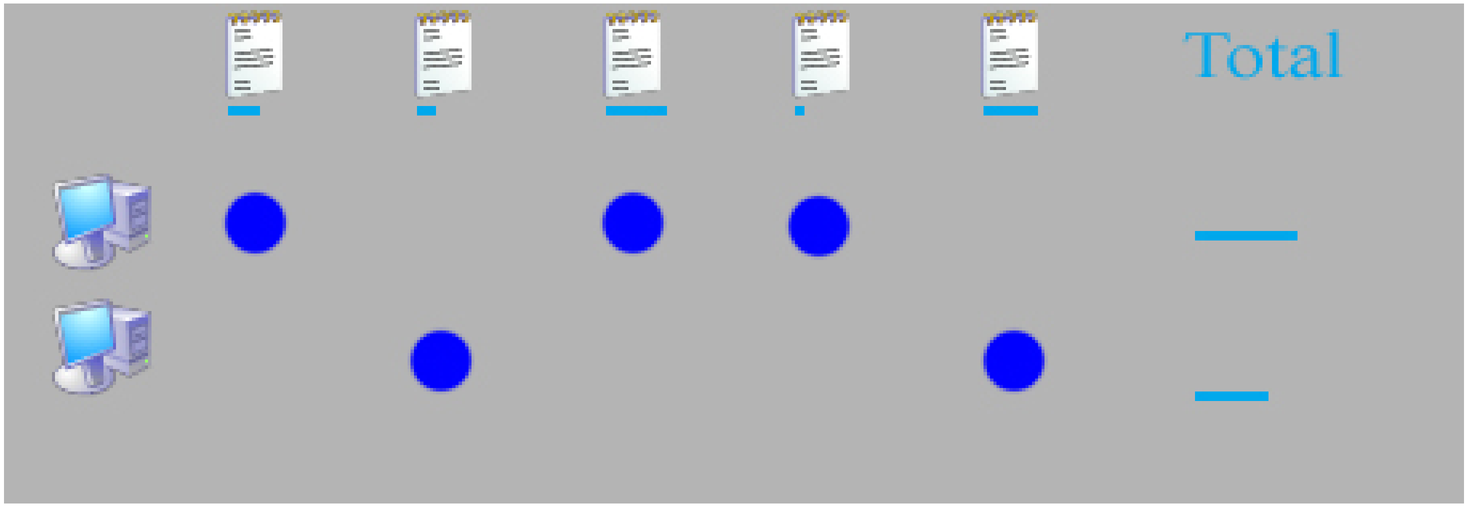
$\nexists$ FPTAS - Beispiel



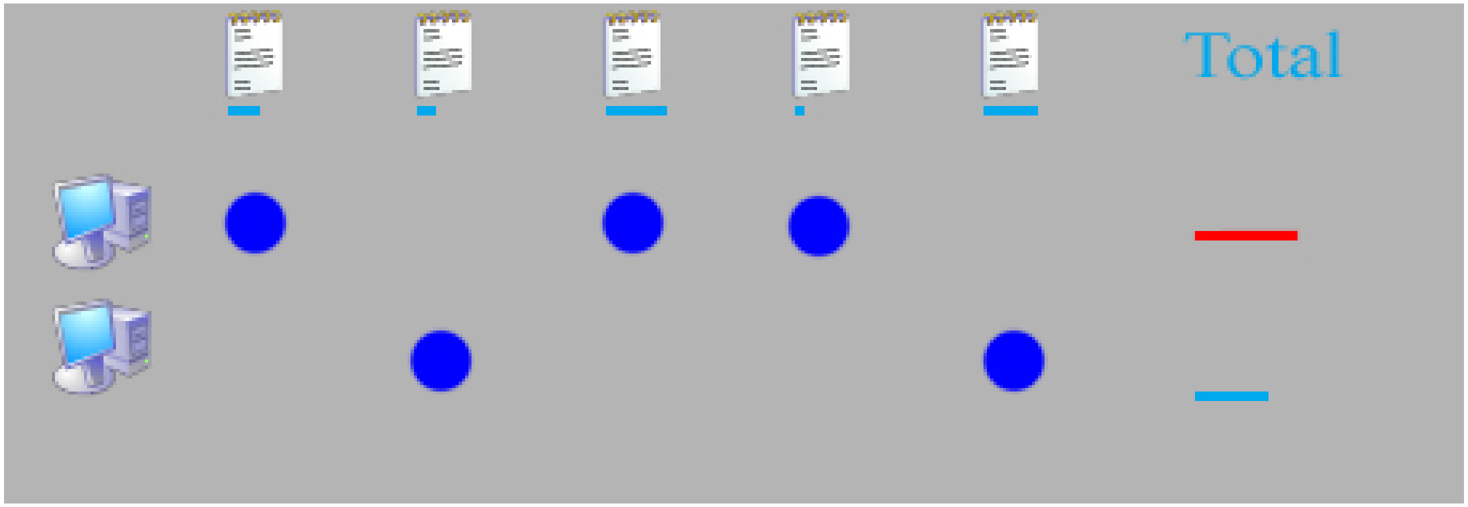
$\nexists$ FPTAS - Beispiel



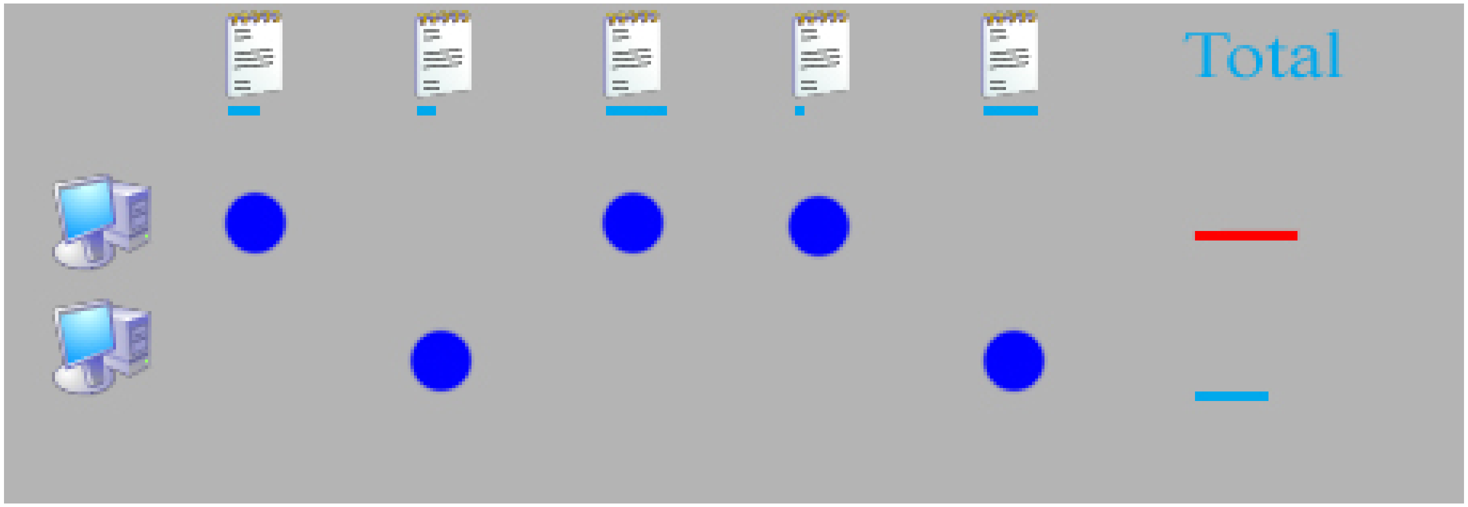
$\nexists$ FPTAS - Beispiel



$\nexists$ FPTAS - Beispiel

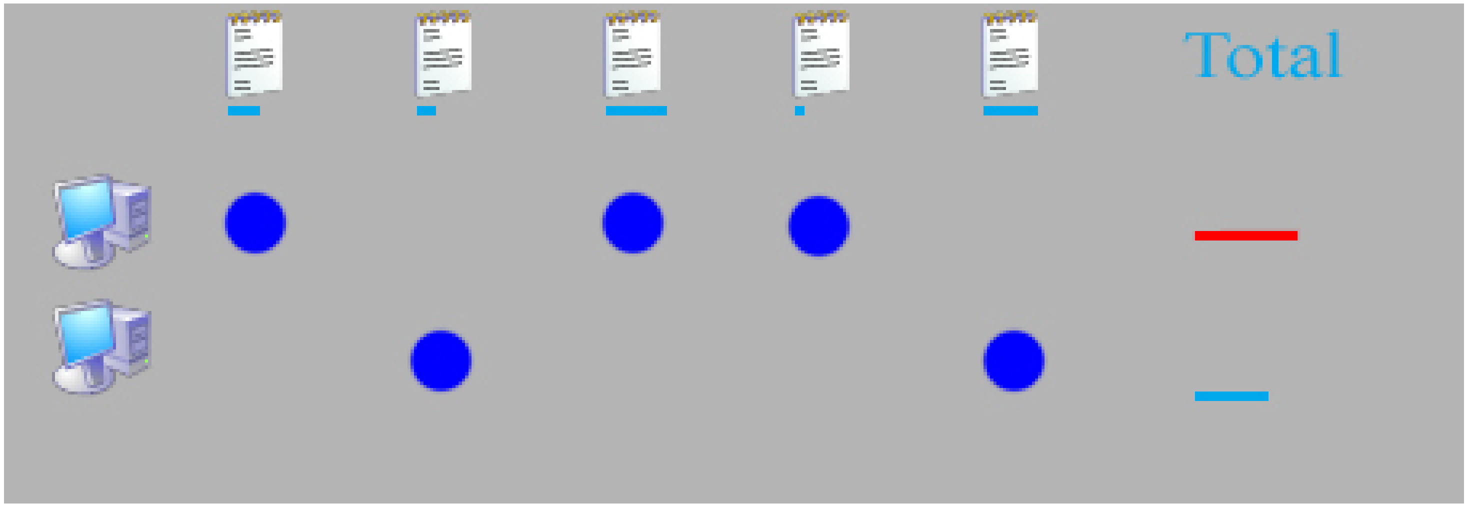


$\nexists$ FPTAS - Beispiel



Z.B. unäre Kodierung:

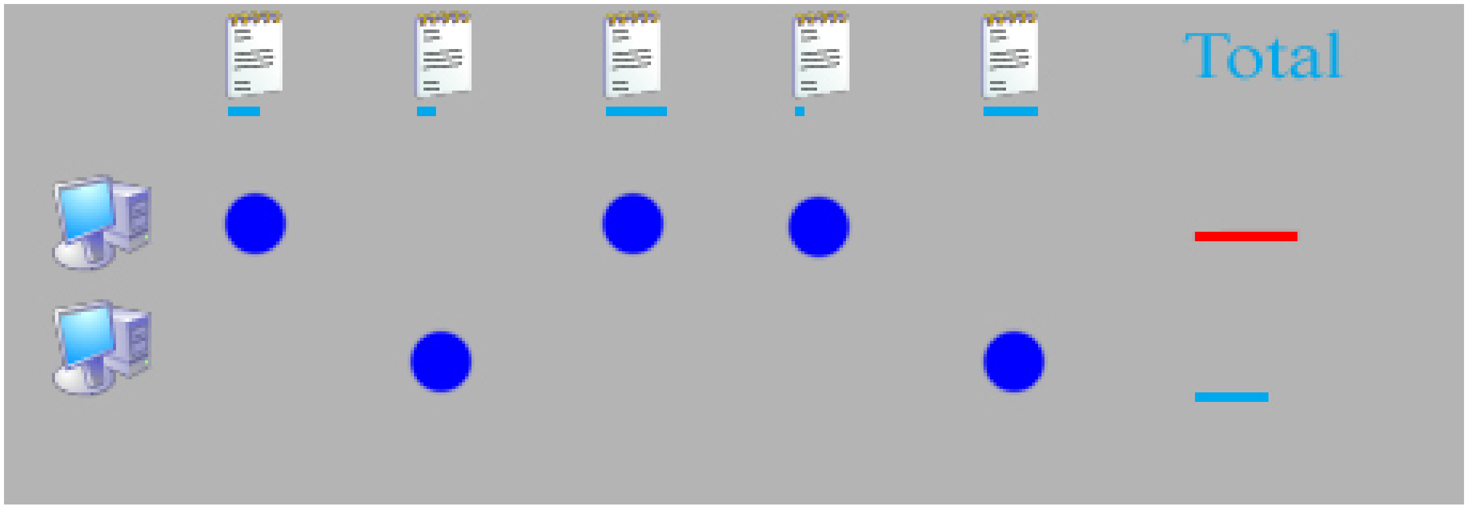
$\nexists$ FPTAS - Beispiel



Z.B. unäre Kodierung:

11\$11111\$111\$11111111\$11\$11111111

$\nexists$ FPTAS - Beispiel

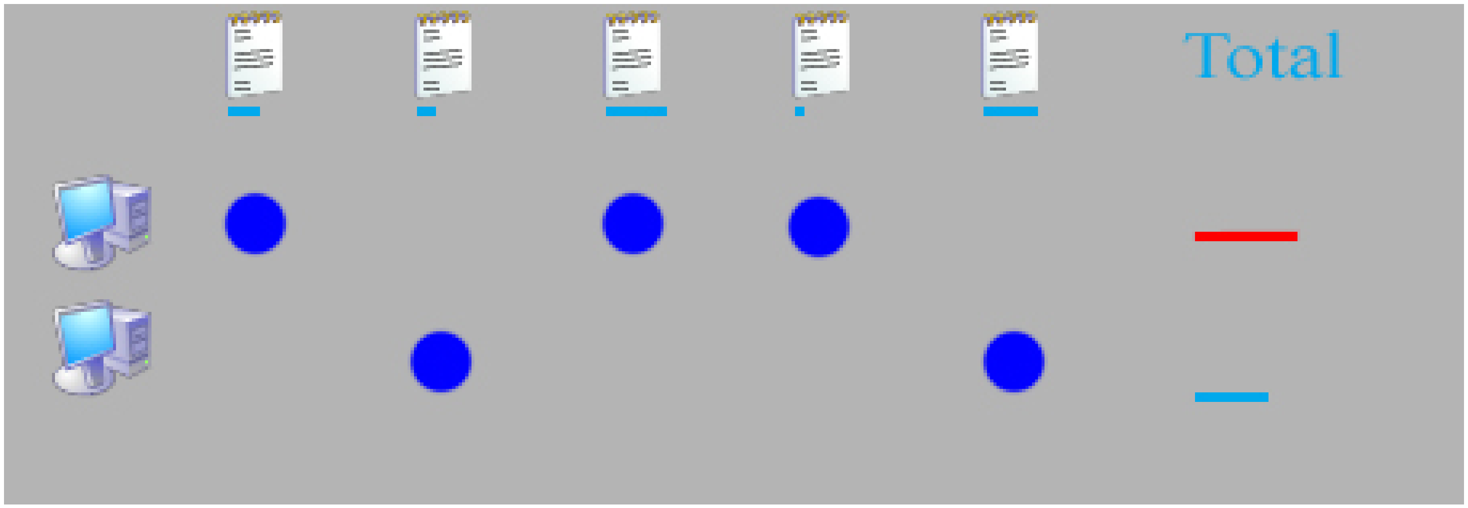


Z.B. unäre Kodierung:

11\$11111\$111\$111111111\$11\$11111111

Benötigte bits: m

$\nexists$ FPTAS - Beispiel

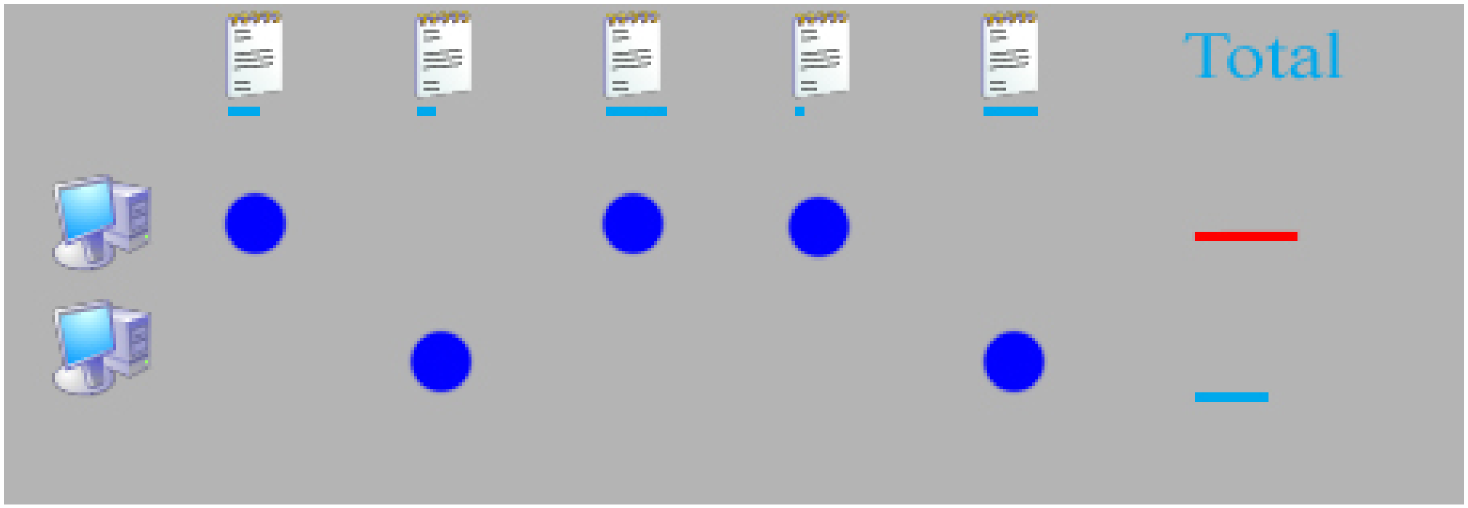


Z.B. unäre Kodierung:

11\$11111\$111\$111111111\$11\$11111111

Benötigte bits: $m + (1 + n - 1)$

$\nexists$ FPTAS - Beispiel

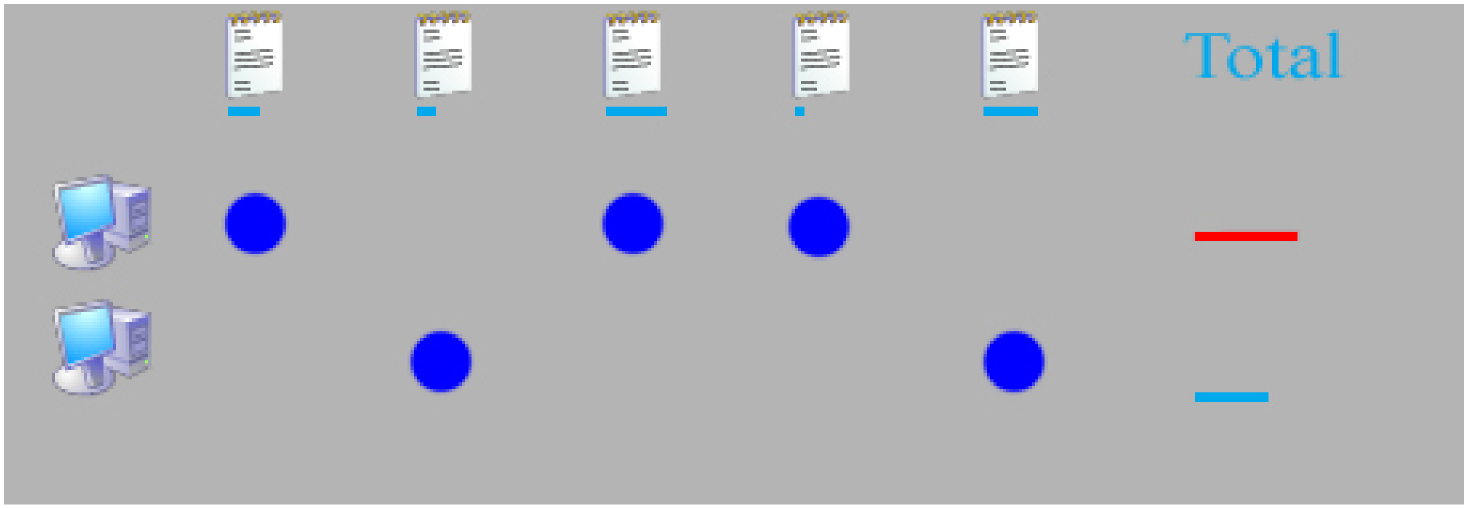


Z.B. unäre Kodierung:

11\$11111\$111\$111111111\$11\$11111111

Benötigte bits: $m + (1 + n - 1) + \underbrace{\sum_{j=1}^{n} p_j}_{p_{\text{sum}}}$

$\nexists$ FPTAS - Beispiel



Z.B. unäre Kodierung:

11\$11111\$111\$111111111\$11\$11111111

Benötigte bits: $m + (1 + n - 1) + \underbrace{\sum_{j=1}^{n} p_j}_{p_{\text{sum}}} = m + n + p_{\text{sum}}$

$\nexists$ FPTAS - Beispiel (2)

Angenommen $P||C_{\max}$ hätte ein FPTAS,
d.h. $P||C_{\max} \in O\left(|I|_{\text{binär}}^{s} \left(\frac{1}{\varepsilon}\right)^{t}\right),\ \ s, t \in \mathbb{R}_{+}$

$\nexists$ FPTAS - Beispiel (2)

Angenommen $P||C_{\max}$ hätte ein FPTAS,
d.h. $P||C_{\max} \in O(|I|^s_{\text{binär}} \left(\frac{1}{\varepsilon}\right)^t), \; s, t \in \mathbb{R}_+$

- Wähle $\varepsilon = (p_{\text{sum}} + 1)^{-1}$.
  FPTAS liefert Lösung in $|I|^s_{\text{binär}}(p_{\text{sum}} + 1)^t$.
  $|I|_{\text{binär}} < |I|_{\text{unär}}, (p_{\text{sum}} + 1)^t \leq (p_{\text{sum}} + m + n)^t = |I|^t_{\text{unär}}$
  $\Rightarrow$ Laufzeit $O(|I|^{s+t}_{\text{unär}})$(polynomiell!).

$\nexists$ FPTAS - Beispiel (2)

Angenommen $P||C_{\max}$ hätte ein FPTAS,
d.h. $P||C_{\max} \in O(|I|^s_{\text{binär}} \left(\frac{1}{\varepsilon}\right)^t)$, $s, t \in \mathbb{R}_+$

- Wähle $\varepsilon = (p_{\text{sum}} + 1)^{-1}$.
  FPTAS liefert Lösung in $|I|^s_{\text{binär}}(p_{\text{sum}} + 1)^t$.
  $|I|_{\text{binär}} < |I|_{\text{unär}}, (p_{\text{sum}} + 1)^t \leq (p_{\text{sum}} + m + n)^t = |I|^t_{\text{unär}}$
  $\Rightarrow$ Laufzeit $O(|I|^{s+t}_{\text{unär}})$(polynomiell!).
- Wie gut ist diese Lösung?
  Angenommen $A(I) \neq \text{OPT}(I)$
  $\text{OPT}(I) + 1 \leq A(I) \leq (1 + \varepsilon)\text{OPT}(I) = (1 + (p_{\text{sum}} + 1)^{-1})\text{OPT}(I) = \text{OPT}(I) + (p_{\text{sum}} + 1)^{-1}\text{OPT}(I)$
  $\Rightarrow \text{OPT}(I) \geq p_{\text{sum}} + 1$ *Widerspruch* $\Rightarrow A(I) = \text{OPT}(I)$

$\nexists$ FPTAS - Beispiel (2)

Angenommen $P||C_{\max}$ hätte ein FPTAS,
d.h. $P||C_{\max} \in O(|I|^s_{\text{binär}} \left(\frac{1}{\varepsilon}\right)^t),\ \ s, t \in \mathbb{R}_+$

- Wähle $\varepsilon = (p_{\text{sum}} + 1)^{-1}$.
  FPTAS liefert Lösung in $|I|^s_{\text{binär}}(p_{\text{sum}} + 1)^t$.
  $|I|_{\text{binär}} < |I|_{\text{unär}}, (p_{\text{sum}} + 1)^t \leq (p_{\text{sum}} + m + n)^t = |I|^t_{\text{unär}}$
  $\Rightarrow$ Laufzeit $O(|I|^{s+t}_{\text{unär}})$(polynomiell!).
- Wie gut ist diese Lösung?
  Angenommen $A(I) \neq \text{OPT}(I)$
  $\text{OPT}(I) + 1 \ \leq \ A(I) \ \leq \ (1 + \varepsilon)\text{OPT}(I) \ = \ (1 + (p_{\text{sum}} + 1)^{-1})\text{OPT}(I) = \text{OPT}(I) + (p_{\text{sum}} + 1)^{-1}\text{OPT}(I)$
  $\Rightarrow \text{OPT}(I) \geq p_{\text{sum}} + 1 \quad$ *Widerspruch* $\Rightarrow A(I) = \text{OPT}(I)$

$\Rightarrow P||C_{\max} \in \text{PP}$ *Widerspruch*

$\nexists$ FPTAS

Satz 1: (gutartig & streng NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X **streng** NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{unär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

$\nexists$ FPTAS

Satz 1: (gutartig & streng NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X **streng** NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{unär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

$\nexists$ FPTAS

Satz 1: (gutartig & streng NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X **streng** NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{unär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

$\nexists$ FPTAS

Satz 1: (gutartig & streng NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X **streng** NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{unär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

$\nexists$ FPTAS

Satz 1: (gutartig & streng NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X **streng** NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{unär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

$\Leftrightarrow (2., 3.\ \wedge \exists$ FPTAS$\Rightarrow$ pseudopolynomiell lösbar).

$\nexists$ FPTAS

Satz 2: (sehr gutartig & NP-schwer $\Rightarrow \nexists$ FPTAS)

1. X NP-schwer.

2. Alle Lösungen haben Kosten $\in \mathbb{Z}$

3. $\textsc{Opt}$ ist durch ein Polynom in $|I|_{\text{binär}}$ beschränkt

$\Rightarrow \nexists$ FPTAS.

Satz 2-Beispiel

Satz 2-Beispiel

2-D Rucksackproblem (ohne „Werte“):

- ▷ $n \in \mathbb{N}$ Gegenstände
- ▷ $v_j \in \mathbb{N}$ Volumen des j-ten Gegenstandes
- ▷ $w_j \in \mathbb{N}$ Gewicht des j-ten Gegenstandes
- ▷ $V \in \mathbb{N}$ Volumenkapazität des Rucksacks
- ▷ $W \in \mathbb{N}$ Gewichtskapazität des Rucksacks

Suche $T \subseteq \{1, \ldots, n\}$, so dass
$\sum_{j \in T} v_j \leq V$, $\sum_{j \in T} w_j \leq W$ und $|T|$ maximal.
pseudopolynomiell lösbar, aber:

Satz 2-Beispiel

2-D Rucksackproblem (ohne „Werte“):

- ▷ $n \in \mathbb{N}$ Gegenstände
- ▷ $v_j \in \mathbb{N}$ Volumen des j-ten Gegenstandes
- ▷ $w_j \in \mathbb{N}$ Gewicht des j-ten Gegenstandes
- ▷ $V \in \mathbb{N}$ Volumenkapazität des Rucksacks
- ▷ $W \in \mathbb{N}$ Gewichtskapazität des Rucksacks

Suche $T \subseteq \{1, \ldots, n\}$, so dass
$\sum_{j \in T} v_j \leq V$, $\sum_{j \in T} w_j \leq W$ und $|T|$ maximal.
pseudopolynomiell lösbar, aber:
NP-schwer

Satz 2-Beispiel

2-D Rucksackproblem (ohne „Werte“):

- ▷ $n \in \mathbb{N}$ Gegenstände
- ▷ $v_j \in \mathbb{N}$ Volumen des j-ten Gegenstandes
- ▷ $w_j \in \mathbb{N}$ Gewicht des j-ten Gegenstandes
- ▷ $V \in \mathbb{N}$ Volumenkapazität des Rucksacks
- ▷ $W \in \mathbb{N}$ Gewichtskapazität des Rucksacks

Suche $T \subseteq \{1, \ldots, n\}$, so dass
$\sum_{j \in T} v_j \leq V,\ \sum_{j \in T} w_j \leq W$ und $|T|$ maximal.
pseudopolynomiell lösbar, aber:
NP-schwer
$\textsc{Opt}(I) \leq p(|I|)$

Satz 2-Beispiel

2-D Rucksackproblem (ohne „Werte“):

- ▷ $n \in \mathbb{N}$ Gegenstände
- ▷ $v_j \in \mathbb{N}$ Volumen des j-ten Gegenstandes
- ▷ $w_j \in \mathbb{N}$ Gewicht des j-ten Gegenstandes
- ▷ $V \in \mathbb{N}$ Volumenkapazität des Rucksacks
- ▷ $W \in \mathbb{N}$ Gewichtskapazität des Rucksacks

Suche $T \subseteq \{1, \ldots, n\}$, so dass
$\sum_{j \in T} v_j \leq V, \ \sum_{j \in T} w_j \leq W$ und $|T|$ maximal.
pseudopolynomiell lösbar, aber:
NP-schwer
$\textsc{Opt}(I) \leq p(|I|) \quad \checkmark$, für $p(X) = X$

Satz 2-Beispiel

2-D Rucksackproblem (ohne „Werte“):

- $n \in \mathbb{N}$ Gegenstände
- $v_j \in \mathbb{N}$ Volumen des j-ten Gegenstandes
- $w_j \in \mathbb{N}$ Gewicht des j-ten Gegenstandes
- $V \in \mathbb{N}$ Volumenkapazität des Rucksacks
- $W \in \mathbb{N}$ Gewichtskapazität des Rucksacks

Suche $T \subseteq \{1, \ldots, n\}$, so dass
$\sum_{j \in T} v_j \leq V,\ \sum_{j \in T} w_j \leq W$ und $|T|$ maximal.
pseudopolynomiell lösbar, aber:
NP-schwer
$\textsc{Opt}(I) \leq p(|I|)$ ✓, für $p(X) = X \Rightarrow \nexists$ FPTAS

$\nexists$ PTAS - Die Lückentechnik

Die Lückentechnik (für Minimierungsprobleme)

Betrachte ganzz. Minimierungsproblem Y. Gegeben:

- ▷ X NP-schweres Entscheidungsproblem
- ▷ Transformation $\tau : X \to Y$ polynomiell berechenbar.
- ▷ $a, b \in \mathbb{N},\ a < b$ fest.
- ▷ $I_X \in X$ JA-Instanz $\Rightarrow \textsc{Opt}(\tau(I_X)) \leq a$
- ▷ $I_X \in X$ NEIN-Instanz $\Rightarrow \textsc{Opt}(\tau(I_X)) \geq b$

$\nexists$ PTAS - Die Lückentechnik

Die Lückentechnik (für Minimierungsprobleme)
Betrachte ganzz. Minimierungsproblem Y. Gegeben:

- $\triangleright$ X NP-schweres Entscheidungsproblem
- $\triangleright$ Transformation $\tau : X \to Y$ polynomiell berechenbar.
- $\triangleright$ $a, b \in \mathbb{N},\ a < b$ fest.
- $\triangleright$ $I_X \in X$ JA-Instanz $\Rightarrow \textsc{Opt}(\tau(I_X)) \leq a$
- $\triangleright$ $I_X \in X$ NEIN-Instanz $\Rightarrow \textsc{Opt}(\tau(I_X)) \geq b$

$\Rightarrow Y$ hat kein ρ-Approximationsalgorithmus mit $\rho < \frac{b}{a}$
Insbesondere hat Y kein PTAS.

$\nexists$ PTAS - Die Lückentechnik

Angenommen $\exists \rho$-Approximationsalgorithmus mit $\rho < \frac{b}{a}$

$I_X \in X, \; I_Y = \tau(I_X) \in Y$

$\nexists$ PTAS - Die Lückentechnik

Angenommen $\exists\rho$-Approximationsalgorithmus mit $\rho < \frac{b}{a}$

$I_X \in X,\ I_Y = \tau(I_X) \in Y$

Falls $c(s_{I_Y}) \geq b$ dann $b \leq c(s_{I_Y}) \leq \rho \cdot \textsc{Opt}(I_Y)$

$< \frac{b}{a} \cdot \textsc{Opt}(I_Y) \Rightarrow 1 < \frac{1}{a} \cdot \textsc{Opt}(I_Y)$

$\Rightarrow a < \textsc{Opt}(I_Y) \Rightarrow I_X$ ist NEIN-Instanz.

Analog: $c(s_{I_Y}) \leq a \Rightarrow I_X$ ist JA-Instanz

$\Rightarrow X$ polynomiell lösbar Widerspruch, falls $\mathrm{P} \neq \mathrm{NP}$

Die Lückentechnik-Lenstras Unmöglichkeitssatz

Lenstras Unmöglichkeitssatz
(Spezialfall der Lückentechnik).

- ▷ Y ganzz. Minimierungsproblem
- ▷ $g \in \mathbb{N}$, fest
- ▷ „Hat Y Lsg. mit Kosten $\leq g$" sei NP-schwer

$\Rightarrow Y$ hat kein ρ-Approximationsalgorithmus mit $\rho < \frac{g+1}{g}$
Insbesondere hat Y kein PTAS.

Lenstras Unmöglichkeitssatz - Beispiel

$Y = R||C_{\max}$, Minimierung der Abarbeitungszeit von n Aufträgen auf m unabhängige Maschinen.

Abarbeitungszeit von Auftrag J_j hängt von Maschine M_i ab, beträgt p_{ij}.

Lenstras Unmöglichkeitssatz - Beispiel

$Y = R||C_{\max}$, Minimierung der Abarbeitungszeit von n Aufträgen auf m unabhängige Maschinen.
Abarbeitungszeit von Auftrag J_j hängt von Maschine M_i ab, beträgt p_{ij}.Beispiel: $n = 7, m = 5$

	M_1	M_2	M_3	M_4	M_5
J_1	4	3	11	5	7
J_2	5	8	2	6	3
J_3	1	5	11	7	2
J_4	12	9	3	8	3
J_5	12	12	9	3	6
J_6	4	8	9	1	6
J_7	8	13	11	11	2

Lenstras Unmöglichkeitssatz - Beispiel

$Y = R||C_{\max}$, Minimierung der Abarbeitungszeit von n Aufträgen auf m unabhängige Maschinen.
Abarbeitungszeit von Auftrag J_j hängt von Maschine M_i ab, beträgt p_{ij}.Beispiel: $n = 7, m = 5$

	M_1	M_2	M_3	M_4	M_5
J_1	4	3	11	5	7
J_2	5	8	2	6	3
J_3	1	5	11	7	2
J_4	12	9	3	8	3
J_5	12	12	9	3	6
J_6	4	8	9	1	6
J_7	8	13	11	11	2

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$
Hilfsmittel: NP-schweres 3-Dimensionale
Zuordnungsproblem (3DZP):

- $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- $T \subseteq A \times B \times C$
- $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$

Hilfsmittel: NP-schweres 3-Dimensionale Zuordnungsproblem (3DZP):

- ▷ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- ▷ $T \subseteq A \times B \times C$
- ▷ $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Polynomielle Reduktion von 3DZP auf $R||C_{\max}$:

$|T| = m$

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$

Hilfsmittel: NP-schweres 3-Dimensionale Zuordnungsproblem (3DZP):

- ▷ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- ▷ $T \subseteq A \times B \times C$
- ▷ $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Polynomielle Reduktion von 3DZP auf $R||C_{\max}$:

$|T| = m \quad ,T_i \ni (a_j, b_k, c_l) \mapsto M_i = M(T_i)$

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$
Hilfsmittel: NP-schweres 3-Dimensionale Zuordnungsproblem (3DZP):

- $\triangleright$ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- $\triangleright$ $T \subseteq A \times B \times C$
- $\triangleright$ $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Polynomielle Reduktion von 3DZP auf $R||C_{\max}$:
$|T| = m \quad ,T_i \ni (a_j, b_k, c_l) \mapsto M_i = M(T_i)$
$A \cup B \cup C \ni x \mapsto$ Elementauftrag $J(x)$. (insg. $3q$)

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$

Hilfsmittel: NP-schweres 3-Dimensionale Zuordnungsproblem (3DZP):

- ▷ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- ▷ $T \subseteq A \times B \times C$
- ▷ $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Polynomielle Reduktion von 3DZP auf $R||C_{\max}$:

$|T| = m \quad ,T_i \ni (a_j, b_k, c_l) \mapsto M_i = M(T_i)$

$A \cup B \cup C \ni x \mapsto$ Elementauftrag $J(x)$. (insg. $3q$)

Zusätzlich: $|T| - q$ Platzhalteraufträge.

Lenstras Unmöglichkeitssatz - Beispiel

$R||C_{\max}$ hat kein ρ-Approximationsalgorithmus für $\rho < \frac{4}{3}$
Hilfsmittel: NP-schweres 3-Dimensionale Zuordnungsproblem (3DZP):

- ▷ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- ▷ $T \subseteq A \times B \times C$
- ▷ $\exists T' \subseteq T,\ |T'| = q$, das $A \cup B \cup C$ überdeckt?

Polynomielle Reduktion von 3DZP auf $R||C_{\max}$:
$|T| = m \quad ,T_i \ni (a_j, b_k, c_l) \mapsto M_i = M(T_i)$
$A \cup B \cup C \ni x \mapsto$ Elementauftrag $J(x)$. (insg. $3q$)
Zusätzlich: $|T| - q$ Platzhalteraufträge.
Abarbeitungszeiten: $x \in T_i,\ J(x)$ auf M_i 1, sonst 3.

Lenstras Unmöglichkeitssatz - Beispiel

T enthällt perfekte Zuordnung

Lenstras Unmöglichkeitssatz - Beispiel

T enthällt perfekte Zuordnung

$\Rightarrow \exists$ Ablaufplannung der Länge 3

Lenstras Unmöglichkeitssatz - Beispiel

T enthällt perfekte Zuordnung
$\Leftrightarrow \exists$ Ablaufplannung der Länge 3

Lenstras Unmöglichkeitssatz - Beispiel

T enthällt perfekte Zuordnung

$\Leftrightarrow \exists$ Ablaufplannung der Länge 3

Da 3DZP NP-schwer ist, ist es auch

„$R||C_{\max}$ hat Lsg. mit Kosten ≤ 3“

Lenstras Unmöglichkeitssatz - Beispiel

T enthällt perfekte Zuordnung
$\Leftrightarrow \exists$ Ablaufplannung der Länge 3
Da 3DZP NP-schwer ist, ist es auch
„$R||C_{\max}$ hat Lsg. mit Kosten ≤ 3“
$\xRightarrow{\text{Lenstra}} \nexists \rho$ - Approx.alg. mit $\rho < \frac{4}{3}$
Insbesondere gibt es kein FPTAS für dieses Problem.

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...

...X NP-schwer.

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...

...X NP-schwer. Bew: Polynomielle Reduktion

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...

...X NP-schwer. Bew: Polynomielle Reduktion

...X APX-schwer.

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...

...X NP-schwer. Bew: Polynomielle Reduktion

...X APX-schwer. Bew: Approximierbarkeitsbewahrende Reduktion.

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...
...X NP-schwer. Bew: Polynomielle Reduktion
...X APX-schwer. Bew: Approximierbarkeitsbewahrende Reduktion.

Es gibt verschiedene solche Reduktionen.

Kein PTAS - APX-Vollständigkeit

Ziel: Zeige dass...
...X NP-schwer. Bew: Polynomielle Reduktion
...X APX-schwer. Bew: Approximierbarkeitsbewahrende Reduktion.

Es gibt verschiedene solche Reduktionen.
Z.B. die L-Reduktion.

Kein PTAS - L-Reduktion

Gegeben: Minimierungsprobleme X, Y.

Kein PTAS - L-Reduktion

Gegeben: Minimierungsprobleme X, Y.

L-**Reduktion:** Paar von Funktionen (R, S):

- $R: \; X \to Y$, polynomiell berechenbar
- $S: \; S_{I_Y} \to S_{I_X}$, polynomiell berechenbar
- $\exists \alpha \in \mathbb{R}^+ \; : \; \textsc{Opt}(R(I_X)) \leq \alpha \cdot \textsc{Opt}(I_X)$
- $\exists \beta \in \mathbb{R}^+ \; : \; |c(S(s_{I_Y})) - \textsc{Opt}(I_X)| \leq \beta \cdot |c(s_{I_Y}) - \textsc{Opt}(R(I_X))|$

Kein PTAS - L-Reduktion

Gegeben: Minimierungsprobleme X, Y.

L-**Reduktion:** Paar von Funktionen (R, S):

- $\triangleright$ $R: \ X \rightarrow Y$, polynomiell berechenbar
- $\triangleright$ $S: \ S_{I_Y} \rightarrow S_{I_X}$, polynomiell berechenbar
- $\triangleright$ $\exists \alpha \in \mathbb{R}^+ \ : \ \textsc{Opt}(R(I_X)) \leq \alpha \cdot \textsc{Opt}(I_X)$
- $\triangleright$ $\exists \beta \in \mathbb{R}^+ \ : \ |c(S(s_{I_Y})) - \textsc{Opt}(I_X)| \leq \beta \cdot |c(s_{I_Y}) - \textsc{Opt}(R(I_X))|$

„$\textsc{Opt}$ wird nicht viel schlechter“

Kein PTAS - L-Reduktion

Gegeben: Minimierungsprobleme X, Y.

L-**Reduktion:** Paar von Funktionen (R, S):

- ▷ $R: \; X \rightarrow Y$, polynomiell berechenbar
- ▷ $S: \; S_{I_Y} \rightarrow S_{I_X}$, polynomiell berechenbar
- ▷ $\exists \alpha \in \mathbb{R}^+ \; : \; \textsc{Opt}(R(I_X)) \leq \alpha \cdot \textsc{Opt}(I_X)$
- ▷ $\exists \beta \in \mathbb{R}^+ \; : \; |c(S(s_{I_Y})) - \textsc{Opt}(I_X)| \leq \beta \cdot |c(s_{I_Y}) - \textsc{Opt}(R(I_X))|$

„$\textsc{Opt}$ wird nicht viel schlechter“

„Man entfernt sich nicht sehr vom $\textsc{Opt}$“

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:
X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:
X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.
Hat Y einen $(1 + \varepsilon)$-AA, so hat X einen $(1 + \alpha\beta\varepsilon)$-AA

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

I_X

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

$$I_X \qquad \xrightarrow{R^{\alpha}}$$

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1 + \varepsilon)$-AA, so hat X einen $(1 + \alpha\beta\varepsilon)$-AA

$$I_X \qquad \xrightarrow{R^{\alpha}} \qquad I_Y$$

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

$$
\begin{array}{ccc}
I_X & \xrightarrow{R^{\alpha}} & I_Y \\
 & & \\
 & & \downarrow (1+\varepsilon)-\mathrm{AA}
\end{array}
$$

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

$$
\begin{array}{ccc}
I_X & \xrightarrow{R^{\alpha}} & I_Y \\
 & & \downarrow (1+\varepsilon)\mathrm{-AA} \\
 & & s_{I_Y}
\end{array}
$$

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

$$\begin{array}{ccc}
I_X & \xrightarrow{R^{\alpha}} & I_Y \\
 & & \downarrow (1+\varepsilon)\mathrm{-AA} \\
 & \xleftarrow{S^{\beta}} & s_{I_Y}
\end{array}$$

Satz: L-Reduktion ist approximierbarkeitsbewahrend

Aus L-Reduktion folgt:

X, Y Minimierungsprobleme, (R, S) eine L-Reduktion mit α, β.

Hat Y einen $(1+\varepsilon)$-AA, so hat X einen $(1+\alpha\beta\varepsilon)$-AA

$$\begin{tikzcd} I_X \arrow[r, "R^{\alpha}"] & I_Y \arrow[d, "(1+\varepsilon)-\mathrm{AA}"] \\ s_{I_X} & s_{I_Y} \arrow[l, "S^{\beta}"'] \end{tikzcd}$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)}$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

$$\leq \beta \frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\frac{\textsc{Opt}(I_Y)}{\alpha}}$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

$$\leq \beta\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\frac{\textsc{Opt}(I_Y)}{\alpha}} = \alpha\beta\underbrace{\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_Y)}}_{\leq\varepsilon}$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

$$\leq \beta\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\frac{\textsc{Opt}(I_Y)}{\alpha}} = \alpha\beta\underbrace{\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_Y)}}_{\leq\varepsilon} \leq \alpha\beta\varepsilon$$

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta |c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

$$\leq \beta \frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\frac{\textsc{Opt}(I_Y)}{\alpha}} = \alpha\beta \underbrace{\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_Y)}}_{\leq \varepsilon} \leq \alpha\beta\varepsilon$$

$\alpha\beta\varepsilon$ ist eine Konstante

L-Reduktion ist approximierbarkeitsbewahrend - Beweis

Beweis:

$$\frac{|c(S(s_{I_Y})) - \textsc{Opt}(I_X)|}{\textsc{Opt}(I_X)} \leq \frac{\beta |c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_X)}$$

$$\leq \beta \frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\frac{\textsc{Opt}(I_Y)}{\alpha}} = \alpha\beta \underbrace{\frac{|c(s_{I_Y}) - \textsc{Opt}(I_Y)|}{\textsc{Opt}(I_Y)}}_{\leq \varepsilon} \leq \alpha\beta\varepsilon$$

$\alpha\beta\varepsilon$ ist eine Konstante

$\Rightarrow$ Die L-Reduktion liefert einen $(1 + \alpha\beta\varepsilon)$ - Approximationsalgorithmus für das Problem X.

L-Reduktion ist approximierbarkeitsbewahrend

Satz:

X, Y Optimierungsprobleme, (R, S) eine L-Reduktion von X nach Y.Dann gilt:

Y besitzt ein PTAS $\Rightarrow X$ besitzt ein PTAS.

X besitzt kein PTAS $\Rightarrow Y$ besitzt kein PTAS.

L-Reduktion ist approximierbarkeitsbewahrend

Satz:
X, Y Optimierungsprobleme, (R, S) eine L-Reduktion von X nach Y.Dann gilt:
Y besitzt ein PTAS $\Rightarrow X$ besitzt ein PTAS.
X besitzt kein PTAS $\Rightarrow Y$ besitzt kein PTAS.

Satz:
APX-schweres Problem hat ein PTAS $\Rightarrow \mathrm{P} = \mathrm{NP}$.

L-Reduktion - Beispiel

Vorbemerkungen:

L-Reduktion - Beispiel

Vorbemerkungen:

MAX-3DM-B: Optimierungsversion des 3D-Zuordnungsproblems:

- $\triangleright$ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- $\triangleright$ $T \subseteq A \times B \times C$
- $\triangleright$ $\forall x \in A \cup B \cup C\ : 1 \leq |\{t \in T :\ x \in t\}| \leq 3$

L-Reduktion - Beispiel

Vorbemerkungen:

MAX-3DM-B: Optimierungsversion des 3D-Zuordnungsproblems:

- $\triangleright$ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- $\triangleright$ $T \subseteq A \times B \times C$
- $\triangleright$ $\forall x \in A \cup B \cup C\ : 1 \leq |\{t \in T :\ x \in t\}| \leq 3$

Finde $T' \subseteq T$ maximaler Kardinalität, so dass je zwei Tripel in allen Komponenten verschieden sind.

L-Reduktion - Beispiel

Vorbemerkungen:

$\textsc{Max-3DM-B}$: Optimierungsversion des 3D-Zuordnungsproblems:

- $\triangleright$ $A = \{a_1, \ldots, a_q\},\ B = \{b_1, \ldots, b_q\},\ C = \{c_1, \ldots, c_q\}$
- $\triangleright$ $T \subseteq A \times B \times C$
- $\triangleright$ $\forall x \in A \cup B \cup C\ : 1 \leq |\{t \in T :\ x \in t\}| \leq 3$

Finde $T' \subseteq T$ maximaler Kardinalität, so dass je zwei Tripel in allen Komponenten verschieden sind.

Spezialfall: Nur Instanzen für die $\textsc{Opt} = q$.

L-Reduktion - Beispiel

Max-3DM-B- Beispiel:

$q = 3$

$A = \{a_1, a_2, a_3\},\ B = \{b_1, b_2, b_3\},\ C = \{c_1, c_2, c_3\}$

L-Reduktion - Beispiel

Max-3DM-B- Beispiel:

$q = 3$

$A = \{a_1, a_2, a_3\},\ B = \{b_1, b_2, b_3\},\ C = \{c_1, c_2, c_3\}$

$T = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3), (a_1, b_1, c_2)\}$

$\forall x \in A \cup B \cup C\ :\ \exists\, t \in T :\ x \in t$

L-Reduktion - Beispiel

Max-3DM-B- Beispiel:

$q = 3$

$A = \{a_1, a_2, a_3\},\ B = \{b_1, b_2, b_3\},\ C = \{c_1, c_2, c_3\}$

$T = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3), (a_1, b_1, c_2)\}$

$\forall x \in A \cup B \cup C\ :\ \exists\, t \in T :\ x \in t$

$T' = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3)\}$,

$T' \subseteq T, |T'| = 3$

L-Reduktion - Beispiel

Max-3DM-B- Beispiel:

$q = 3$

$A = \{a_1, a_2, a_3\},\ B = \{b_1, b_2, b_3\},\ C = \{c_1, c_2, c_3\}$

$T = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3), (a_1, b_1, c_2)\}$

$\forall x \in A \cup B \cup C\ :\ \exists\, t \in T :\ x \in t$

$T' = \{(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3)\}$,

$T' \subseteq T, |T'| = 3$

$T'' = \{(a_1, b_1, c_2), (a_3, b_3, c_3)\}$,

$T'' \subseteq T,\ |T''| = 2$

L-Reduktion - Beispiel

Vorbemerkungen:

$R||\sum L_i^2$: Ablaufplannung von n Aufträgen auf m unabhängige Maschinen.

L-Reduktion - Beispiel

Vorbemerkungen:

$R||\sum L_i^2$: Ablaufplannung von n Aufträgen auf m unabhängige Maschinen.

Abarbeitungszeit von Auftrag J_j auf Maschine M_i ist p_{ij}.

Minimiere $\sum L_i^2$, wobei L_i die Belastung („Arbeitszeit“) von M_i.

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.
Bekannt: $\textsc{Max}$-3DM-B ist APX-schwer.

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.

Bekannt: $\textsc{Max}$-$3\mathrm{DM}$-B ist APX-schwer.

Es genügt eine L-Reduktion von $\textsc{Max}$-$3\mathrm{DM}$-B auf $R||\sum L_i^2$

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.

Bekannt: $\textsc{Max}\text{-}3\mathrm{DM}\text{-}\mathrm{B}$ ist APX-schwer.

Es genügt eine L-Reduktion von $\textsc{Max}\text{-}3\mathrm{DM}\text{-}\mathrm{B}$ auf $R||\sum L_i^2$

Vorgehensweise:

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.
Bekannt: $\textsc{Max}$-3DM-B ist APX-schwer.
Es genügt eine L-Reduktion von $\textsc{Max}$-3DM-B auf $R||\sum L_i^2$

Vorgehensweise:

- Suche geeignete Funktionen R, S

L-Reduktion - Beispiel

Ziel: $R||\sum L_i^2$ ist APX-schwer.
Bekannt: $\textsc{Max}$-3DM-B ist APX-schwer.
Es genügt eine L-Reduktion von $\textsc{Max}$-3DM-B auf $R||\sum L_i^2$

Vorgehensweise:

- Suche geeignete Funktionen R, S
- Überprüfe ob es α, β gibt, für die die Bedingungen erfüllt werden.

R : **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$|A| = |B| = |C| = q \;\rightsquigarrow\; m = 3q,\; n = 5q$$

R : **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$
\begin{array}{lll}
|A| = |B| = |C| = q & \rightsquigarrow & m = 3q,\ n = 5q \\
t \in T & \rightsquigarrow & M(t) \\
 & & 3q - |T| \text{ Platzhaltermaschinen}
\end{array}
$$

R : **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$
\begin{array}{lcl}
|A| = |B| = |C| = q & \rightsquigarrow & m = 3q,\ n = 5q \\
t \in T & \rightsquigarrow & M(t) \\
 & & 3q - |T| \text{ Platzhaltermaschinen} \\
a_i \in A \cup B \cup C & \rightsquigarrow & J(a_i) \\
 & & 2q \text{ Platzhalteraufträge}
\end{array}
$$

R : **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$
\begin{array}{lcl}
|A| = |B| = |C| = q & \rightsquigarrow & m = 3q,\ n = 5q \\
t \in T & \rightsquigarrow & M(t) \\
 & & 3q - |T| \text{ Platzhaltermaschinen} \\
a_i \in A \cup B \cup C & \rightsquigarrow & J(a_i) \\
 & & 2q \text{ Platzhalteraufträge}
\end{array}
$$

Abarbeitungszeiten: $(x \in t,\ M_i = M(t))$

	M_i	$M_j,\ j \neq i$
$J(x)$	1	∞
Platzhalterauftrag	3	3

R : **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$
\begin{array}{lcl}
|A| = |B| = |C| = q & \rightsquigarrow & m = 3q,\ n = 5q \\
t \in T & \rightsquigarrow & M(t) \\
 & & 3q - |T| \text{ Platzhaltermaschinen} \\
a_i \in A \cup B \cup C & \rightsquigarrow & J(a_i) \\
 & & 2q \text{ Platzhalteraufträge}
\end{array}
$$

Abarbeitungszeiten: ($x \in t,\ M_i = M(t)$)

	M_i	$M_j,\ j \neq i$
$J(x)$	1	∞
Platzhalterauftrag	3	3

Polynomiell berechenbar?

$R:$ **Max-3DM-B** $\longrightarrow R||\sum L_i^2$

$$
\begin{array}{lll}
|A| = |B| = |C| = q & \rightsquigarrow & m = 3q,\ n = 5q \\
t \in T & \rightsquigarrow & M(t) \\
 & & 3q - |T| \text{ Platzhaltermaschinen} \\
a_i \in A \cup B \cup C & \rightsquigarrow & J(a_i) \\
 & & 2q \text{ Platzhalteraufträge}
\end{array}
$$

Abarbeitungszeiten: $(x \in t,\ M_i = M(t))$

	M_i	$M_j,\ j \neq i$
$J(x)$	1	∞
Platzhalterauftrag	3	3

Polynomiell berechenbar?✓

$S: \ S_{R||\sum L_i^2} \longrightarrow S_{\textbf{Max-3DM-B}}$

Gegeben: Ablaufplannung s.
Die Maschine M_i heißt gut in s, wenn sie 3 Aufträge der Länge 1 abarbeitet (das können nur „ihre eigene“ Aufträge sein).

$S: \; S_{R||\sum L_i^2} \longrightarrow S_{\textbf{Max-3DM-B}}$

Gegeben: Ablaufplannung s.
Die Maschine M_i heißt gut in s, wenn sie 3 Aufträge der Länge 1 abarbeitet (das können nur „ihre eigene“ Aufträge sein).
Fasse die drei Aufträge einer guten Maschine zu einem Tripel zusammen.
Die Menge aller solcher Tripel sei $S(s_{R||\sum L_i^2})$.

$S: \; S_{R||\sum L_i^2} \longrightarrow S_{\textbf{Max-3DM-B}}$

Gegeben: Ablaufplannung s.
Die Maschine M_i heißt *gut* in s, wenn sie 3 Aufträge der Länge 1 abarbeitet (das können nur „ihre eigene“ Aufträge sein).
Fasse die drei Aufträge einer guten Maschine zu einem Tripel zusammen.
Die Menge aller solcher Tripel sei $S(s_{R||\sum L_i^2})$.
Polynomiell berechenbar?

$S: \; S_{R||\sum L_i^2} \longrightarrow S_{\textbf{Max-3DM-B}}$

Gegeben: Ablaufplannung s.

Die Maschine M_i heißt gut in s, wenn sie 3 Aufträge der Länge 1 abarbeitet (das können nur „ihre eigene“ Aufträge sein).

Fasse die drei Aufträge einer guten Maschine zu einem Tripel zusammen.

Die Menge aller solcher Tripel sei $S(s_{R||\sum L_i^2})$.

Polynomiell berechenbar?✓

$$\textbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \textbf{Opt}(I_{\textbf{Max-3DM-B}})$$

$\textbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \textbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.

$\mathbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \mathbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

$\mathbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \mathbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

- Für $(a_j, b_k, c_l) = t \in \textsc{Opt}(I_{\textsc{Max-3DM-B}})$ teile $J(a_j)$,$J(b_k)$,$J(c_l)$ $M(t)$ zu.

$\mathbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \mathbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

- Für $(a_j, b_k, c_l) = t \in \textsc{Opt}(I_{\textsc{Max-3DM-B}})$ teile $J(a_j)$,$J(b_k)$,$J(c_l)$ $M(t)$ zu.
- Teile den $2q$ Platzhaltermaschinen je einen Platzhalterauftrag zu.

$\textbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \textbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

- ▷ Für $(a_j, b_k, c_l) = t \in \textsc{Opt}(I_{\textsc{Max-3DM-B}})$ teile $J(a_j)$,$J(b_k)$,$J(c_l)$ $M(t)$ zu.
- ▷ Teile den $2q$ Platzhaltermaschinen je einen Platzhalterauftrag zu.

$\Rightarrow$ Alle Maschinen haben Belastung $L_i = 3$ und damit $c(s') = \sum_{i=1}^{3q} L_i^2 = 3q \cdot 3^2 = 27q$.

$\mathbf{Opt}(I_{R||\sum L_i^2}) \leq \alpha \cdot \mathbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

- ▷ Für $(a_j, b_k, c_l) = t \in \textsc{Opt}(I_{\textsc{Max-3DM-B}})$ teile $J(a_j)$,$J(b_k)$,$J(c_l)$ $M(t)$ zu.
- ▷ Teile den $2q$ Platzhaltermaschinen je einen Platzhalterauftrag zu.

$\Rightarrow$ Alle Maschinen haben Belastung $L_i = 3$ und damit
$c(s') = \sum_{i=1}^{3q} L_i^2 = 3q \cdot 3^2 = 27q$.
$\textsc{Opt}(R(I_{\textsc{Max-3DM-B}})) \leq 27q = 27\textsc{Opt}(I_{\textsc{Max-3DM-B}})$

$\mathbf{Opt}(I_{R\|\sum L_i^2}) \leq \alpha \cdot \mathbf{Opt}(I_{\textbf{Max-3DM-B}})$

Es gilt $\textsc{Opt}(I_{\textsc{Max-3DM-B}}) = q$.
Man betrachte die Ablaufplanung s':

- Für $(a_j, b_k, c_l) = t \in \textsc{Opt}(I_{\textsc{Max-3DM-B}})$ teile $J(a_j)$,$J(b_k)$,$J(c_l)$ $M(t)$ zu.
- Teile den $2q$ Platzhaltermaschinen je einen Platzhalterauftrag zu.

$\Rightarrow$ Alle Maschinen haben Belastung $L_i = 3$ und damit
$c(s') = \sum_{i=1}^{3q} L_i^2 = 3q \cdot 3^2 = 27q$.
$\textsc{Opt}(R(I_{\textsc{Max-3DM-B}})) \leq 27q = 27\textsc{Opt}(I_{\textsc{Max-3DM-B}})$
Bedingung erfüllt für $\alpha = 27$.

$$|c(S(s_{R||\sum L_i^2})) - \mathbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \mathbf{Opt}(I_{R||\sum L_i^2})|$$

$$\left|c(S(s_{R||\sum L_i^2})) - \textbf{Opt}(I_{\textbf{Max-3DM-B}})\right| \leq \beta \cdot \left|c(s_{R||\sum L_i^2}) - \textbf{Opt}(I_{R||\sum L_i^2})\right|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:

$$|c(S(s_{R||\sum L_i^2})) - \textbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \textbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:✓

$$|c(S(s_{R||\sum L_i^2})) - \textbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \textbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:✓ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$

$$|c(S(s_{R||\sum L_i^2})) - \mathbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq$$
$$\beta \cdot |c(s_{R||\sum L_i^2}) - \mathbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:✓ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$
(Erinnerung: $c(S(s_{R||\sum L_i^2})) = m_3$)

$$|c(S(s_{R||\sum L_i^2})) - \mathbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq$$
$$\beta \cdot |c(s_{R||\sum L_i^2}) - \mathbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:$\checkmark$ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$
(Erinnerung: $c(S(s_{R||\sum L_i^2})) = m_3$)
$3q = m_0 + m_1 + m_2 + m_3$

$$|c(S(s_{R||\sum L_i^2})) - \mathbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \mathbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:$\checkmark$ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$
(Erinnerung: $c(S(s_{R||\sum L_i^2})) = m_3$)
$3q = m_0 + m_1 + m_2 + m_3$
Die Anzahl der Aufträge der Länge 1 sind:
$3q = m_1 + 2m_2 + 3m_3$

$$|c(S(s_{R||\sum L_i^2})) - \textbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq$$
$$\beta \cdot |c(s_{R||\sum L_i^2}) - \textbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:$\checkmark$ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$
(Erinnerung: $c(S(s_{R||\sum L_i^2})) = m_3$)
$3q = m_0 + m_1 + m_2 + m_3$
Die Anzahl der Aufträge der Länge 1 sind:
$3q = m_1 + 2m_2 + 3m_3$
Später: $c(s) \geq 29q - 2m_3$. Damit:

$$|c(S(s_{R||\sum L_i^2})) - \textbf{Opt}(I_{\textbf{Max-3DM-B}})| \leq$$
$$\beta \cdot |c(s_{R||\sum L_i^2}) - \textbf{Opt}(I_{R||\sum L_i^2})|$$

Falls $\exists$ Auftrag mit Abarbeitungszeit ∞:$\checkmark$ Sonst:
Sei $m_k = \#$ Maschinen die k Aufträge der Länge 1 abarbeiten, $k \in \{0, 1, 2, 3\}$
(Erinnerung: $c(S(s_{R||\sum L_i^2})) = m_3$)
$3q = m_0 + m_1 + m_2 + m_3$
Die Anzahl der Aufträge der Länge 1 sind:
$3q = m_1 + 2m_2 + 3m_3$
Später: $c(s) \geq 29q - 2m_3$. Damit:
$|c(S(s_{R||\sum L_i^2})) - \textsc{Opt}(I_{\textsc{Max-3DM-B}})| = q - m_3 = \frac{1}{2}(29q - 2m_3 - 27q) = \frac{1}{2}|c(s_{R||\sum L_i^2}) - \textsc{Opt}(R(I_{\textsc{Max-3DM-B}}))|$.

und die Bedingung ist für $\beta = \frac{1}{2}$ erfüllt.

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu $\rightsquigarrow s'$.

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu $\rightsquigarrow s'$.
Da $c(s) \geq c(s')$ genügt es zu zeigen, dass $c(s') \geq 29q - 2m_3$.

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu $\rightsquigarrow s'$.
Da $c(s) \geq c(s')$ genügt es zu zeigen, dass
$c(s') \geq 29q - 2m_3$.

- ▷ Jede Maschine höchstens einen Auftrag.

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu $\rightsquigarrow s'$.
Da $c(s) \geq c(s')$ genügt es zu zeigen, dass $c(s') \geq 29q - 2m_3$.

- $\triangleright$ Jede Maschine höchstens einen Auftrag.
- $\triangleright$ Maschinen mit niedriger Belastung bevorzugen

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Man entferne die Platzhalteraufträge aus s und füge sie in günstigster Weise wieder hinzu $\rightsquigarrow s'$.
Da $c(s) \geq c(s')$ genügt es zu zeigen, dass
$c(s') \geq 29q - 2m_3$.

- $\triangleright$ Jede Maschine höchstens einen Auftrag.
- $\triangleright$ Maschinen mit niedriger Belastung bevorzugen

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Zwei Fälle:

- ▷ Maschinen mit Belastung ≤ 1 reichen aus um alle Platzhalteraufträge aufzunehmen.
- ▷ Maschinen mit Belastung ≤ 2 reichen aus um alle Platzhalteraufträge aufzunehmen

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Zwei Fälle:

- ▷ Maschinen mit Belastung ≤ 1 reichen aus um alle Platzhalteraufträge aufzunehmen.
- ▷ Maschinen mit Belastung ≤ 2 reichen aus um alle Platzhalteraufträge aufzunehmen (und die mit Belastung ≤ 1 nicht).

Noch zu zeigen: $c(s) \geq 29q - 2m_3$

Zwei Fälle:

- ▷ Maschinen mit Belastung ≤ 1 reichen aus um alle Platzhalteraufträge aufzunehmen.
- ▷ Maschinen mit Belastung ≤ 2 reichen aus um alle Platzhalteraufträge aufzunehmen (und die mit Belastung ≤ 1 nicht).

In beiden Fällen kann gezeigt werden, dass
$c(s) \geq 29q - 2m_3$ □

L-**Reduktion** - $\ddot{U}bersicht$

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.

L-**Reduktion** - $\ddot{U}bersicht$

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.

Führe L-Reduktion $\textsc{Max}$-3DM-B darauf

L-Reduktion - $\ddot{U}bersicht$

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.

Führe L-Reduktion $\textsc{Max}$-3DM-B darauf

- Finde R um aus einer Instanz von $\textsc{Max}$-3DM-B eine von $R||\sum L_i^2$ zu konstruieren

L-Reduktion - *Übersicht*

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.

Führe L-Reduktion $\textsc{Max}$-3DM-B darauf

- ▷ Finde R um aus einer Instanz von $\textsc{Max}$-3DM-B eine von $R||\sum L_i^2$ zu konstruieren
- ▷ Finde S um aus einer Lösung von $R||\sum L_i^2$ eine von $\textsc{Max}$-3DM-B zu konstruieren

L-Reduktion - *Übersicht*

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.
Führe L-Reduktion Max-3DM-B darauf

- ▷ Finde R um aus einer Instanz von Max-3DM-B eine von $R||\sum L_i^2$ zu konstruieren
- ▷ Finde S um aus einer Lösung von $R||\sum L_i^2$ eine von Max-3DM-B zu konstruieren
- ▷ Prüfe ob es ein $\alpha \in \mathbb{N}$ gibt das $\text{Opt}(R(I_{\text{Max-3DM-B}})) \leq \alpha \cdot \text{Opt}(I_{R||\sum L_i^2})$ erfüllt

L-**Reduktion** - *Übersicht*

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.
Führe L-Reduktion $\textsc{Max-3DM-B}$ darauf

- Finde R um aus einer Instanz von $\textsc{Max-3DM-B}$ eine von $R||\sum L_i^2$ zu konstruieren
- Finde S um aus einer Lösung von $R||\sum L_i^2$ eine von $\textsc{Max-3DM-B}$ zu konstruieren
- Prüfe ob es ein $\alpha \in \mathbb{N}$ gibt das $\textsc{Opt}(R(I_{\textsc{Max-3DM-B}})) \leq \alpha \cdot \textsc{Opt}(I_{R||\sum L_i^2})$ erfüllt
- Prüfe ob es ein $\beta \in \mathbb{N}$ gibt das $|s(S(s_{R||\sum L_i^2})) - \textsc{Opt}(I_{\textsc{Max-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \textsc{Opt}(R(I_{\textsc{Max-3DM-B}}))|$

L-Reduktion - *Übersicht*

Ziel: Zeige dass $R||\sum L_i^2$ APX-schwer ist.
Führe L-Reduktion $\textsc{Max}\text{-3DM-B}$ darauf

- ▷ Finde R um aus einer Instanz von $\textsc{Max}\text{-3DM-B}$ eine von $R||\sum L_i^2$ zu konstruieren
- ▷ Finde S um aus einer Lösung von $R||\sum L_i^2$ eine von $\textsc{Max}\text{-3DM-B}$ zu konstruieren
- ▷ Prüfe ob es ein $\alpha \in \mathbb{N}$ gibt das $\textsc{Opt}(R(I_{\textsc{Max}\text{-3DM-B}})) \leq \alpha \cdot \textsc{Opt}(I_{R||\sum L_i^2})$ erfüllt
- ▷ Prüfe ob es ein $\beta \in \mathbb{N}$ gibt das $|s(S(s_{R||\sum L_i^2})) - \textsc{Opt}(I_{\textsc{Max}\text{-3DM-B}})| \leq \beta \cdot |c(s_{R||\sum L_i^2}) - \textsc{Opt}(R(I_{\textsc{Max}\text{-3DM-B}}))|$

$\Rightarrow$ auch $R||\sum L_i^2$ ist APX-schwer.

Zusammenfassung

Widerlegung der Existenz eines AS

- kein FPTAS
  - *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- kein PTAS
  - die Lückentechnik
  - APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- ▷ kein FPTAS
  - ◇ *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - ◇ *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- ▷ kein PTAS
  - ◇ die Lückentechnik
  - ◇ APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- kein FPTAS
  - *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- kein PTAS
  - die Lückentechnik
  - APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- ▷ kein FPTAS
  - ◇ $gutartig$ und $streng$ NP-schwer $\Rightarrow \nexists$ FPTAS
  - ◇ $sehr\ gutartig$ und NP-schwer $\Rightarrow \nexists$ FPTAS
- ▷ kein PTAS
  - ◇ die Lückentechnik
  - ◇ APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- ▷ kein FPTAS
  - ◇ *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - ◇ *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- ▷ kein PTAS
  - ◇ die Lückentechnik
  - ◇ APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- ▷ kein FPTAS
  - ◇ *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - ◇ *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- ▷ kein PTAS
  - ◇ die Lückentechnik
  - ◇ APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- ▷ kein FPTAS
  - ◇ *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - ◇ *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- ▷ kein PTAS
  - ◇ die Lückentechnik
  - ◇ APX-Vollständigkeit, L-Reduktion

Zusammenfassung

Widerlegung der Existenz eines AS

- kein FPTAS
  - *gutartig* und *streng* NP-schwer $\Rightarrow \nexists$ FPTAS
  - *sehr gutartig* und NP-schwer $\Rightarrow \nexists$ FPTAS
- kein PTAS
  - die Lückentechnik
  - APX-Vollständigkeit, L-Reduktion

Fragen?

Danke für Ihre Aufmerksamkeit!